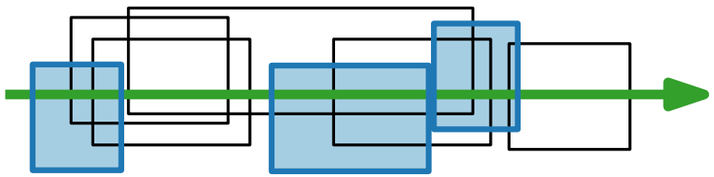
Universität Trier

Algorithmen für geographische Informationssysteme

6. Vorlesung
Kartenbeschriftung



Teil I:
Das Beschriftungsproblem

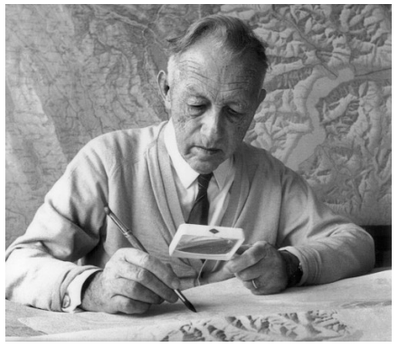
Philipp Kindermann

Wintersemester 2024/25

Kartenbeschriftungen

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz

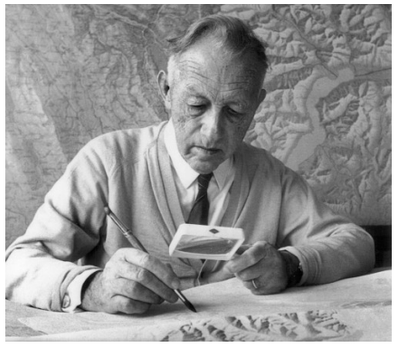


Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kartenbeschriftungen

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



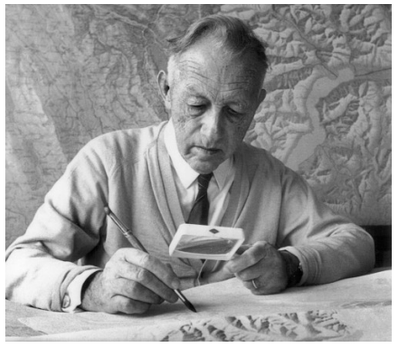
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kartenbeschriftungen

www.worldofmaps.net

"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading."
[E. Imhof '75]

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

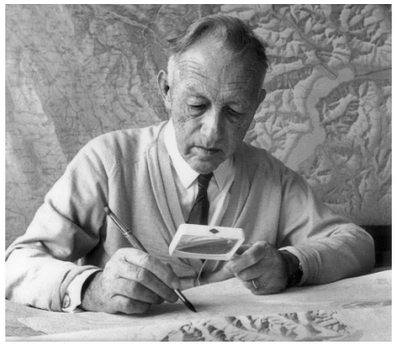
Kartenbeschriftungen

www.worldofmaps.net

"Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading."
[E. Imhof '75]

Beschriftungen für:

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kartenbeschriftungen

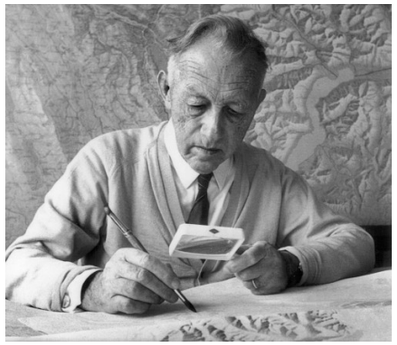
www.worldofmaps.net

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Beschriftungen für:

- Fläche

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

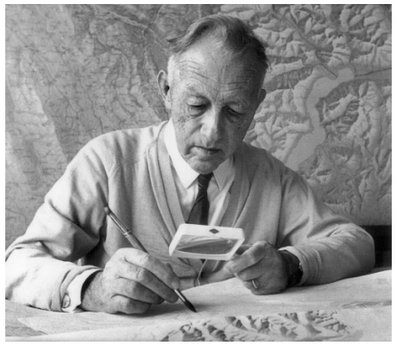
Kartenbeschriftungen

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Beschriftungen für:

- Fläche
- Linien

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

www.worldofmaps.net

Kartenbeschriftungen

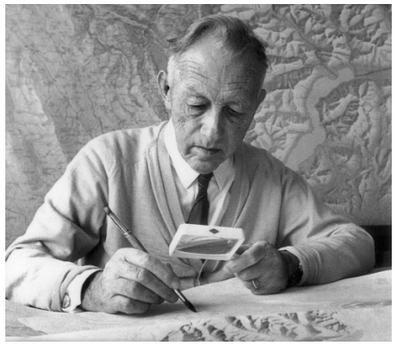
www.worldofmaps.net

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Beschriftungen für:

- Fläche
- Linien
- Punkte

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

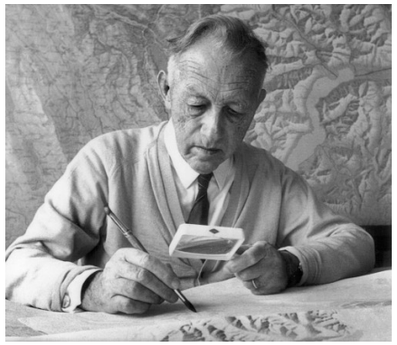
Kartenbeschriftungen

”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Beschriftungen für:

- Fläche
- Linien
- Punkte

Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

www.worldofmaps.net

Kartenbeschriftungen

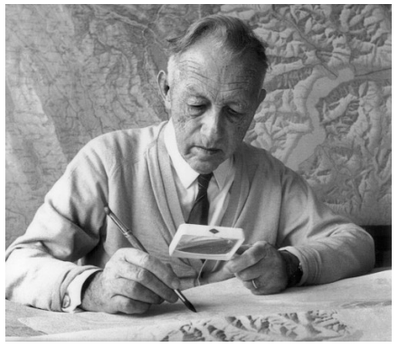
”Poor, sloppy, amateurisch type placement is irresponsible; it spoils even the best image and impedes reading.”
[E. Imhof ’75]

Beschriftungen für:

- Fläche
- Linien
- Punkte

abhängig vom Maßstab!
(Berlin: Fläche oder Punkt)

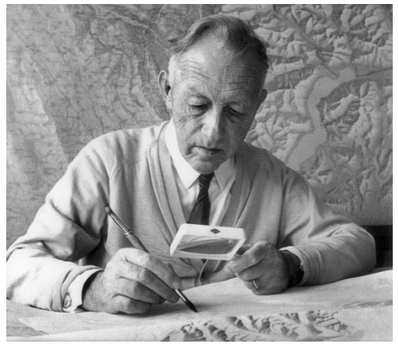
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

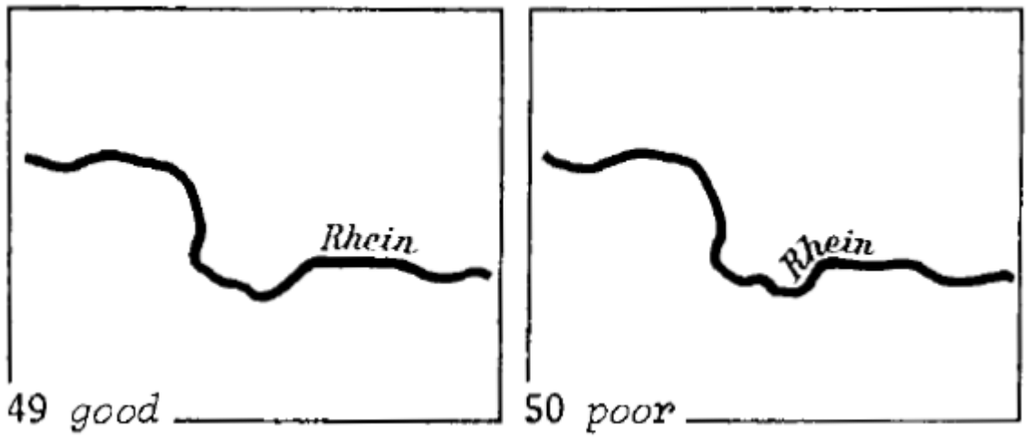
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



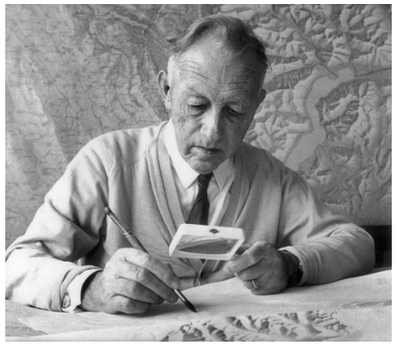
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

- Lesbarkeit



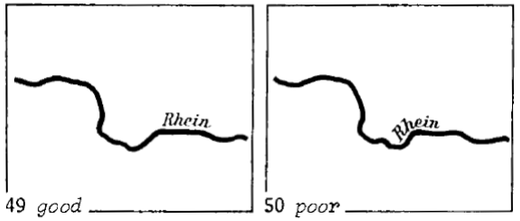
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



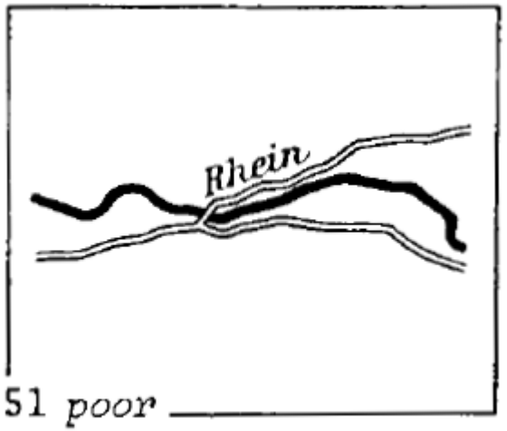
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

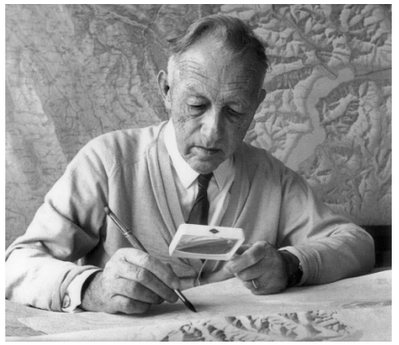
- Lesbarkeit



- klare Zuordnung von Namen zu Objekten



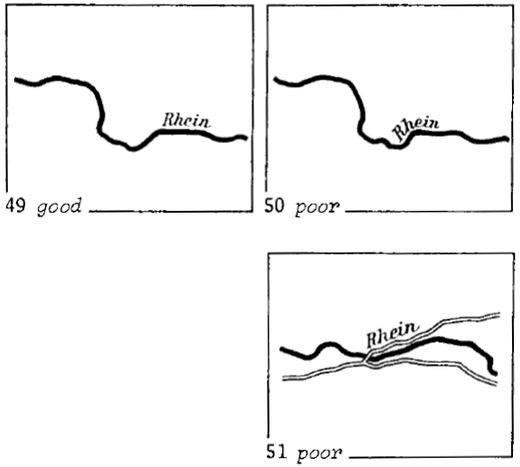
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

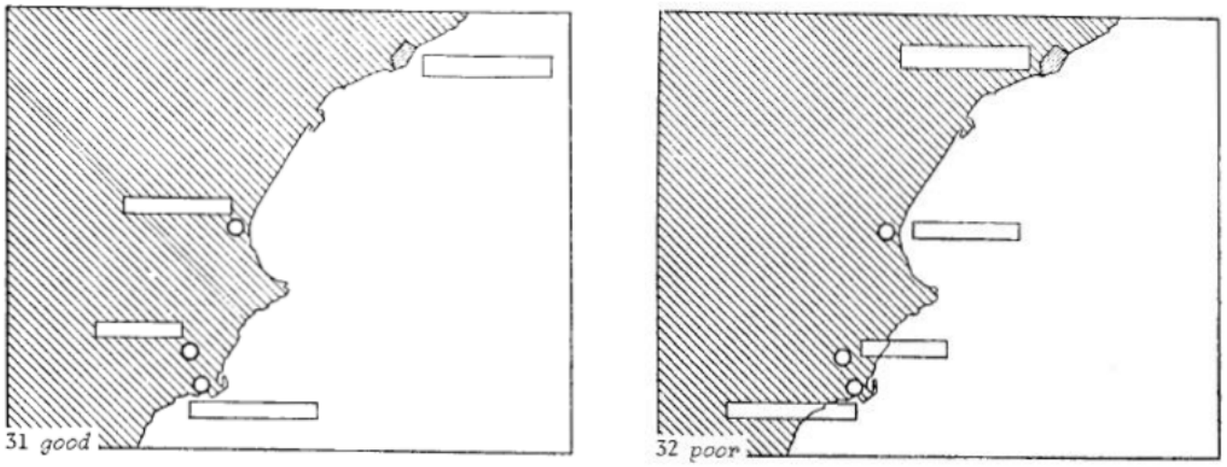
Kriterien für Beschriftungen

- Lesbarkeit

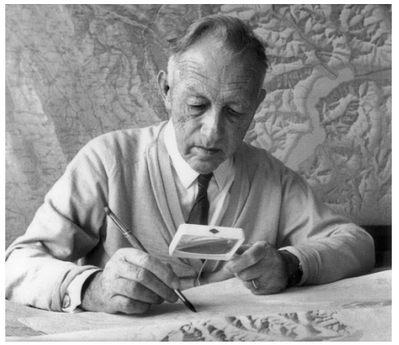


- klare Zuordnung von Namen zu Objekten

- Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)



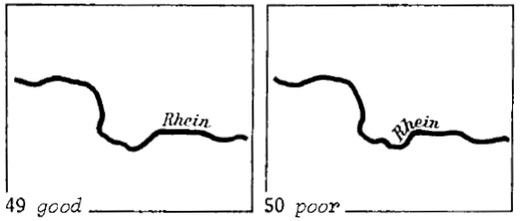
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



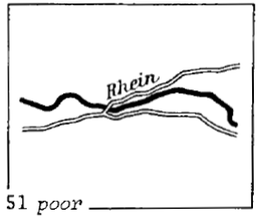
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

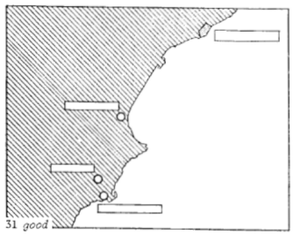
- Lesbarkeit

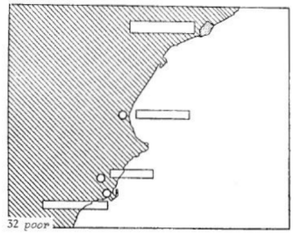


- klare Zuordnung von Namen zu Objekten



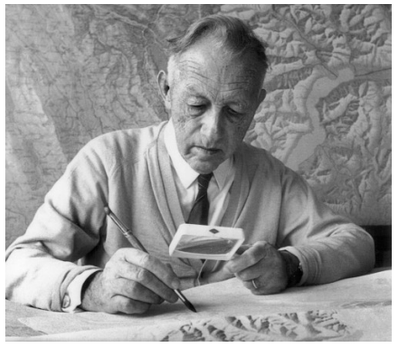
- Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)





- Namen sollen Verständnis von Kartenobjekten erleichtern

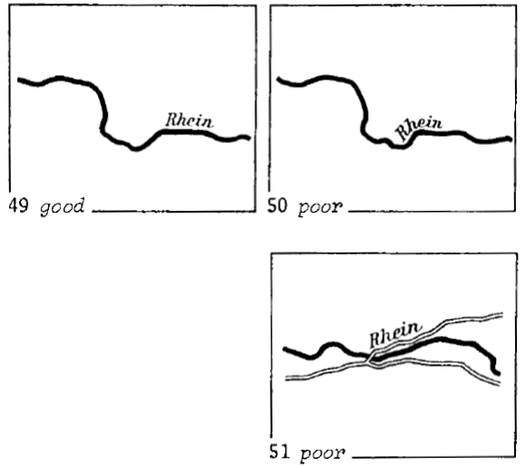
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz

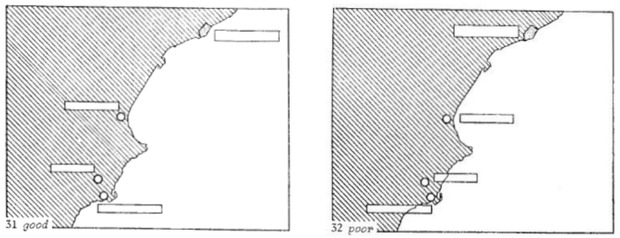


Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

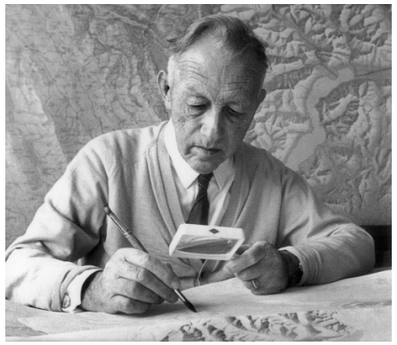
Kriterien für Beschriftungen

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

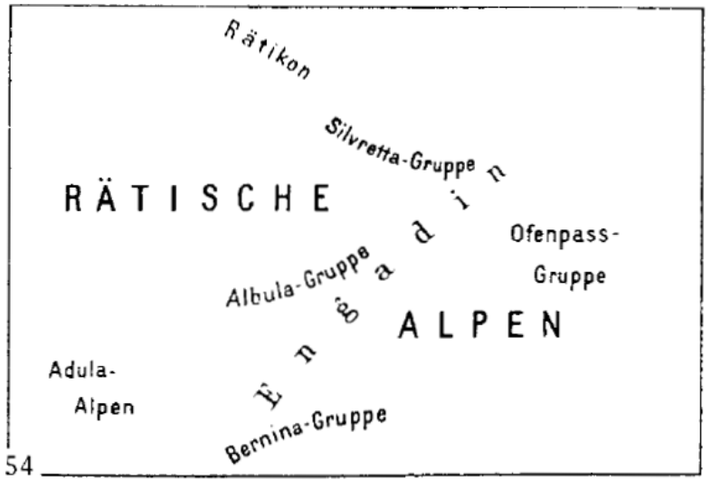




Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz

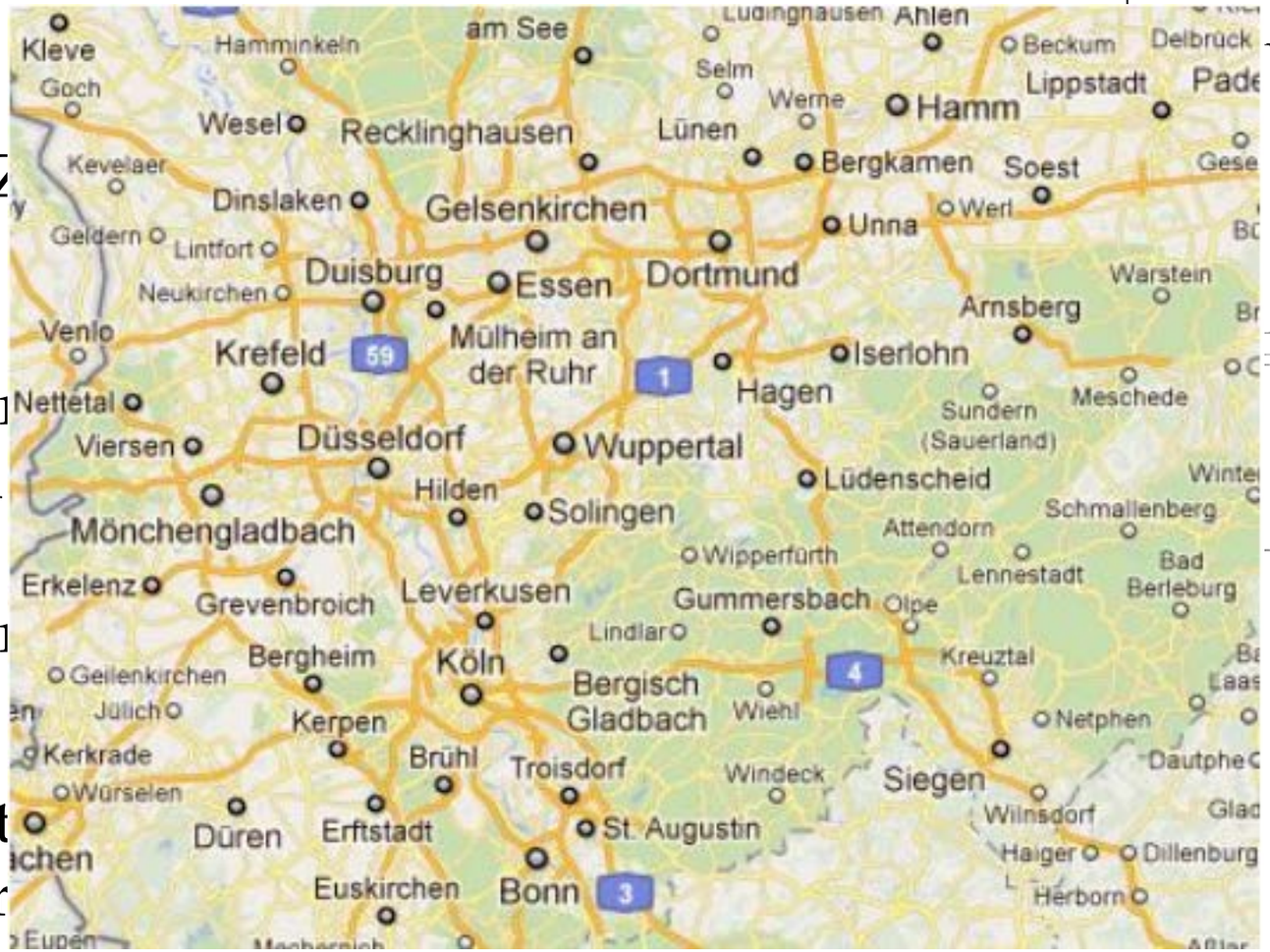


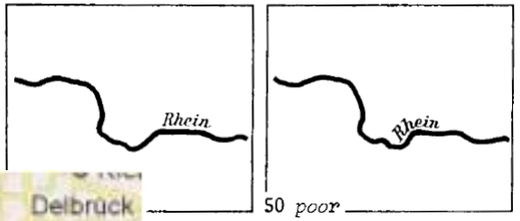
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

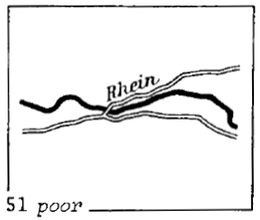


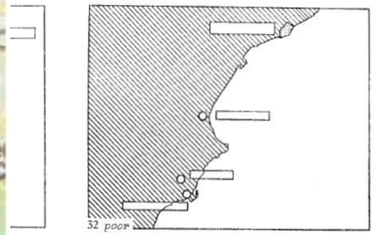
Kriterien für Beschriftungen

- Lesbarkeit
- klare Z
- Name stören
- Name jekten
- Schrift Hierar
- Dichte der Namen soll angemessen variieren.

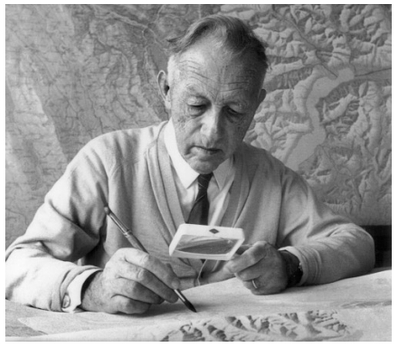








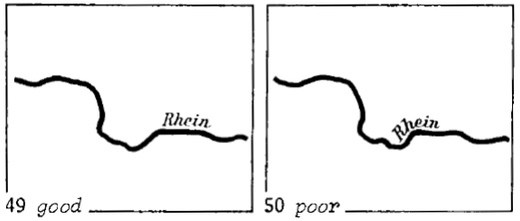
Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



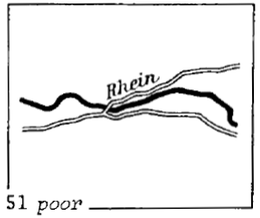
Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Kriterien für Beschriftungen

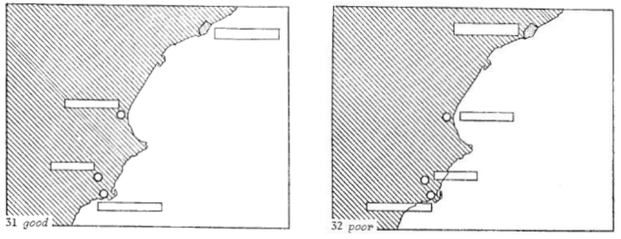
- Lesbarkeit



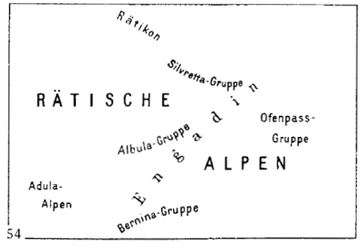
- klare Zuordnung von Namen zu Objekten



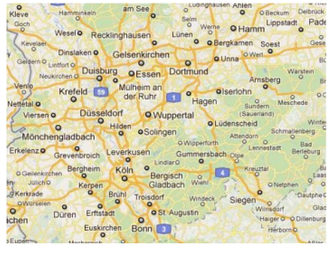
- Namen sollen anderen Karteninhalt wenig stören (keine Verdeckung von Objekten)



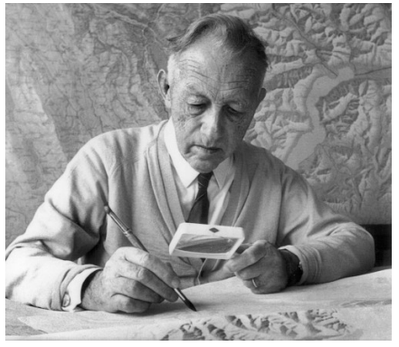
- Namen sollen Verständnis von Kartenobjekten erleichtern



- Schrifttyp und -größe sollen Klassen und Hierarchien wiedergeben.
- Dichte der Namen soll angemessen variieren.



Eduard Imhof
*1895 Schiers, Schweiz
†1986 Erlenbach, Schweiz



Fotografie von Ernst Liniger, um 1980 (Alpines Museum der Schweiz, Bern)

Was können wir anpassen?

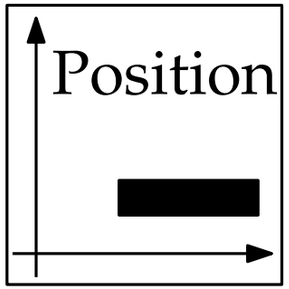
Was können wir anpassen?

Jacques Bertin definierte visuelle Variablen [Bertin'67]

Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

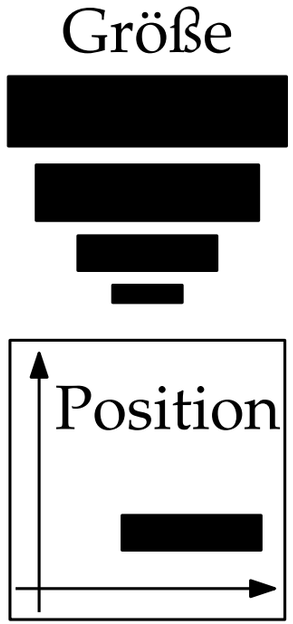
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

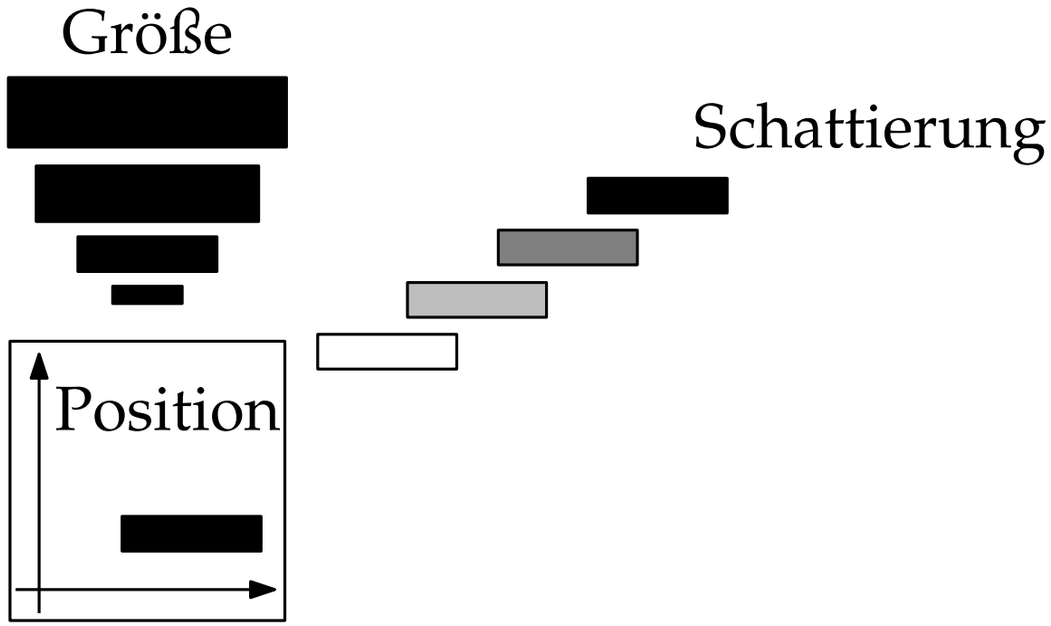
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

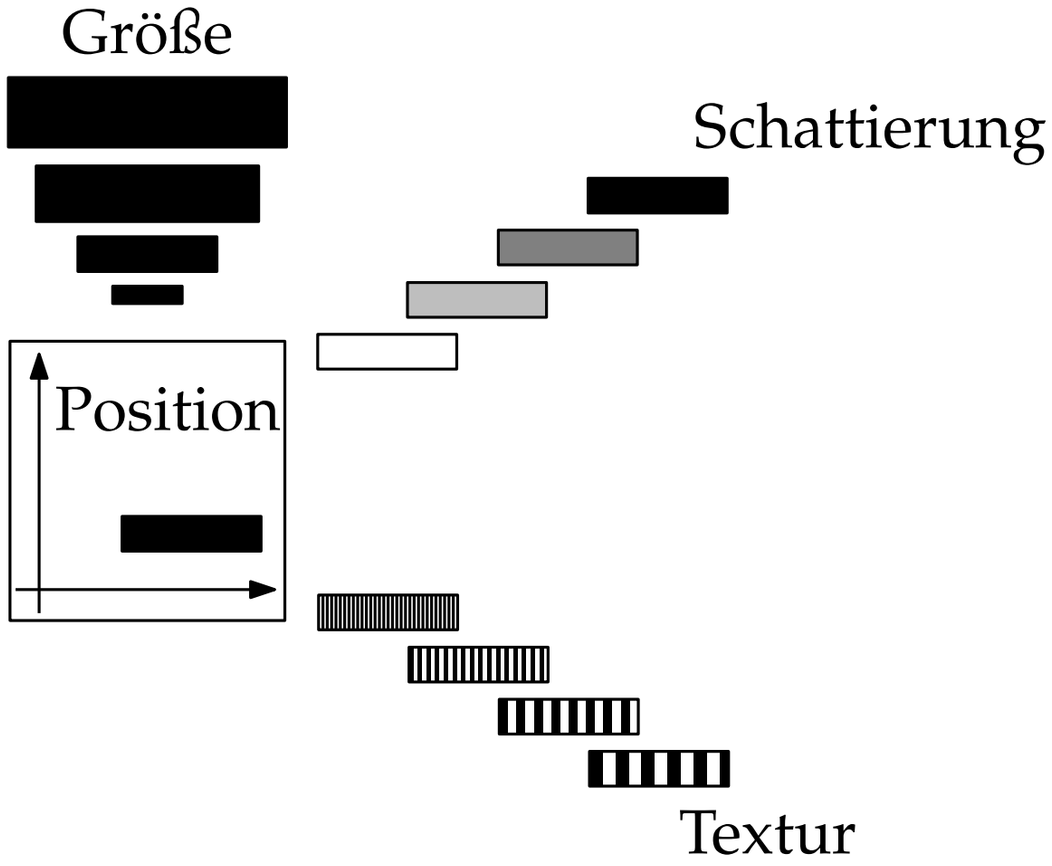
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

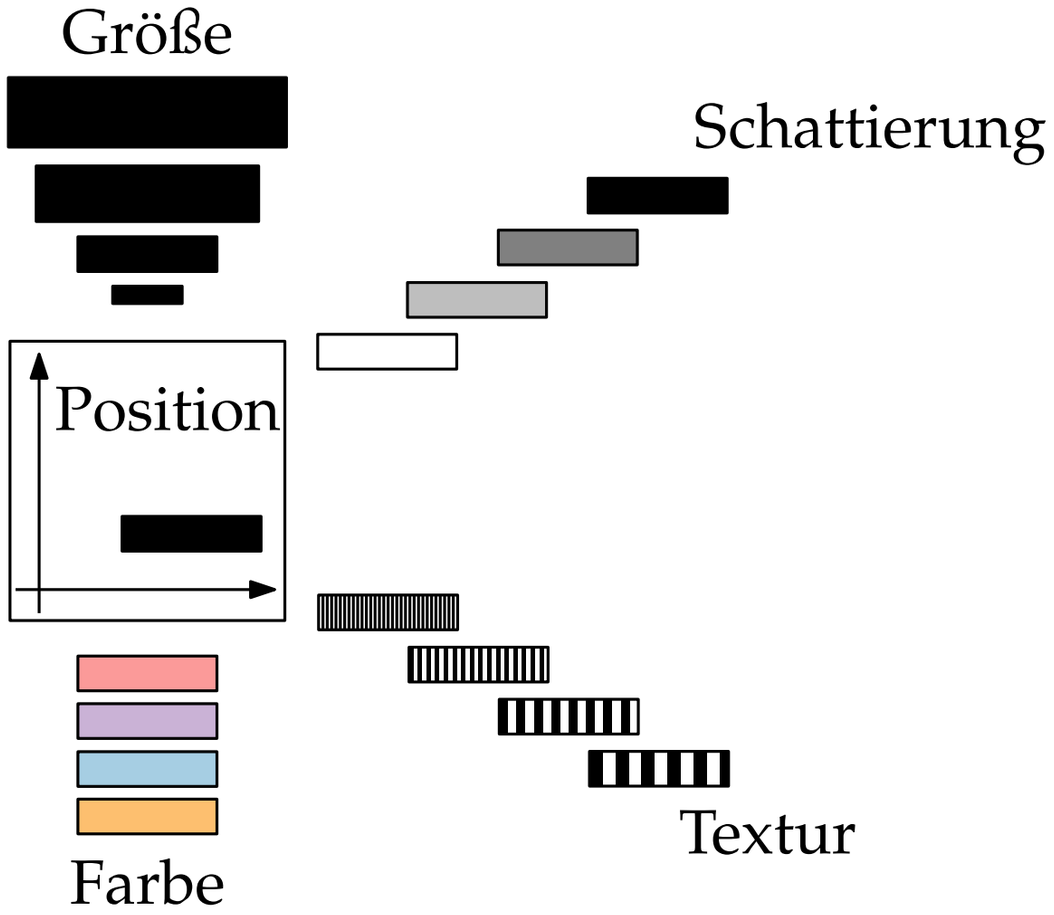
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

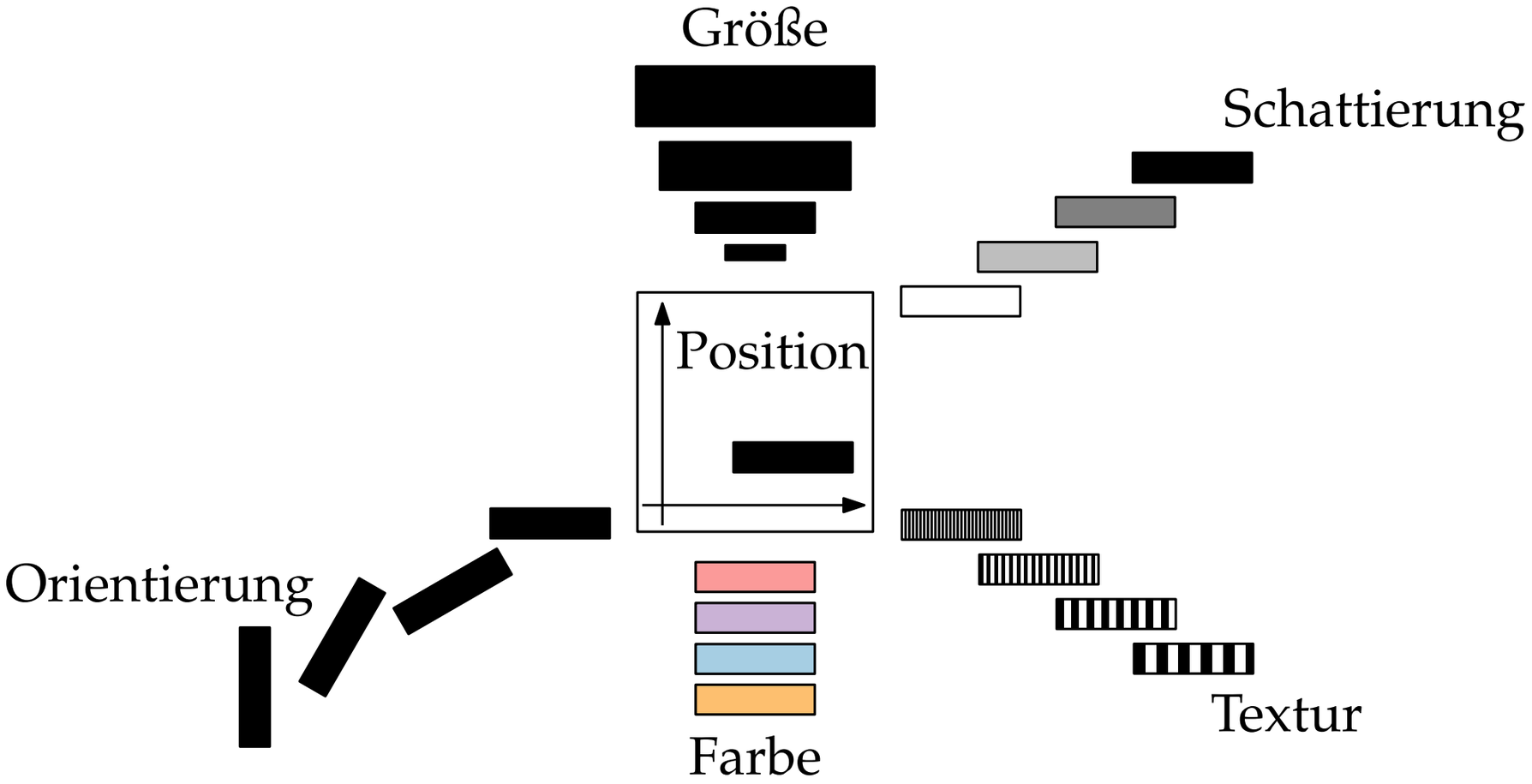
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

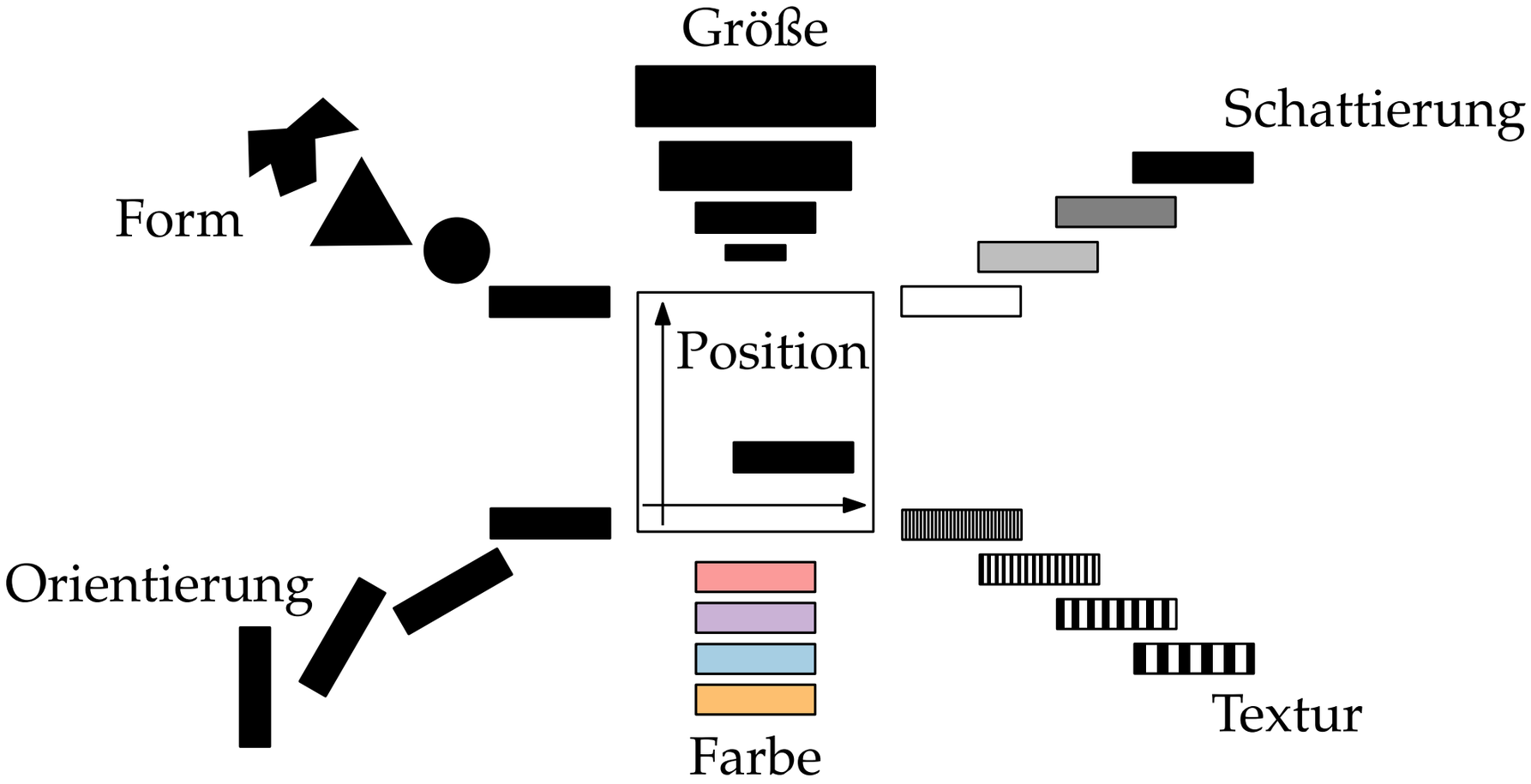
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

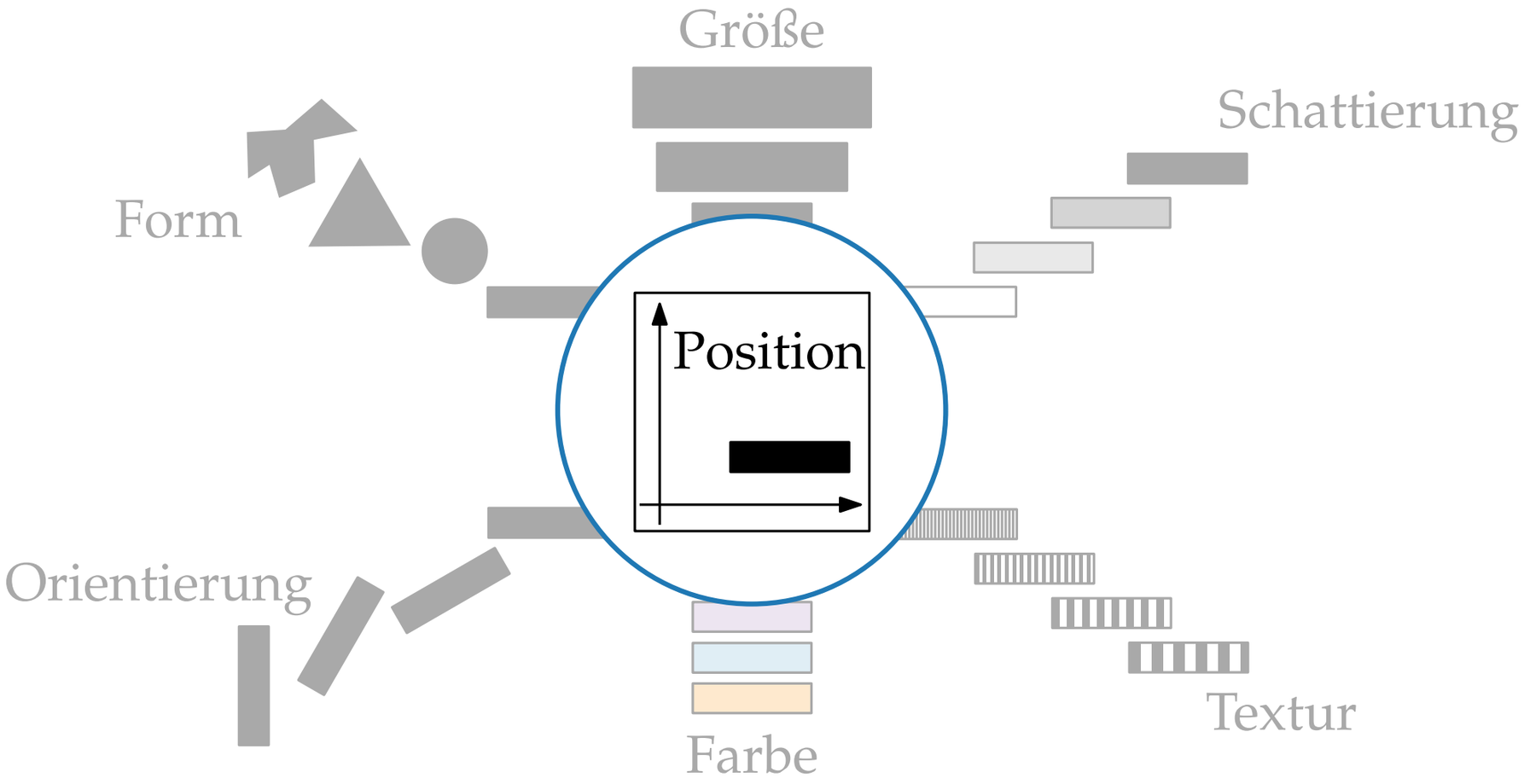
Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Was können wir anpassen?

Jacques Bertin definierte visuelle Variablen [Bertin'67]



Jacques Bertin
*1918 Maisons-Laffitte, Frankreich
†2010 Paris, Frankreich

Geometrische Beschriftungsmodelle

Geg.:

Ges.:

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene

Ges.:

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label,

Ges.:

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.:

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Was ist eine zulässige Beschriftung?

Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Was ist eine zulässige Beschriftung?

diskrete Modelle **Slider-Modelle**

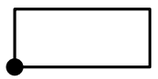
Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Was ist eine zulässige Beschriftung?

diskrete Modelle



1P

Slider-Modelle

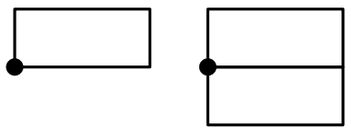
Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Was ist eine zulässige Beschriftung?

diskrete Modelle



1P 2P

Slider-Modelle

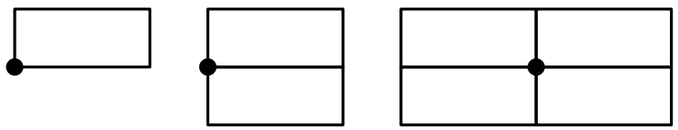
Geometrische Beschriftungsmodelle

Geg.: *n* Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

Was ist eine zulässige Beschriftung?

diskrete Modelle



Slider-Modelle

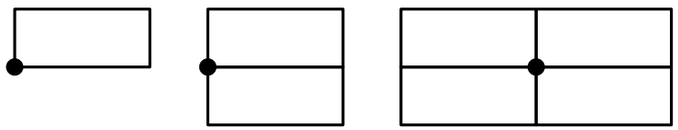
Geometrische Beschriftungsmodelle

Geg.: *n* Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

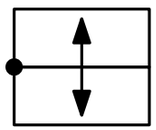
Was ist eine zulässige Beschriftung?

diskrete Modelle



1P 2P 4P

Slider-Modelle



1S

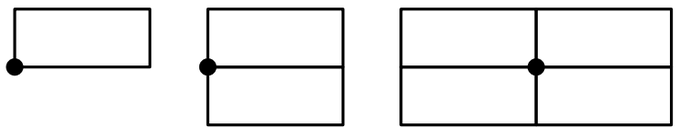
Geometrische Beschriftungsmodelle

Geg.: *n* Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

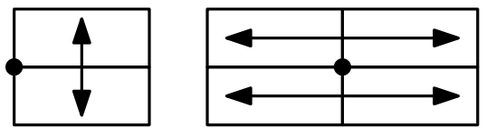
Was ist eine zulässige Beschriftung?

diskrete Modelle



1P 2P 4P

Slider-Modelle



1S 2S

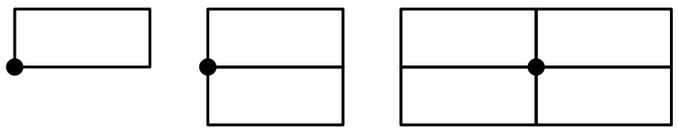
Geometrische Beschriftungsmodelle

Geg.: *n* Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung

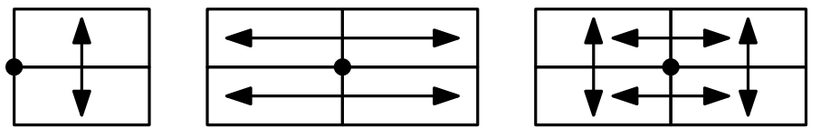
Was ist eine zulässige Beschriftung?

diskrete Modelle



1P 2P 4P

Slider-Modelle



1S 2S 4S

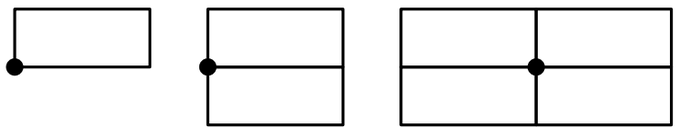
Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

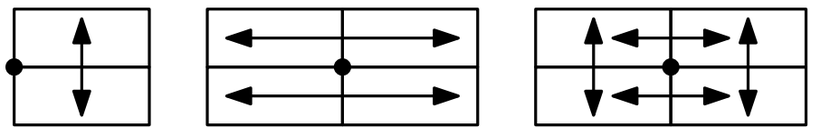
Was ist eine zulässige Beschriftung?

diskrete Modelle



1P 2P 4P

Slider-Modelle

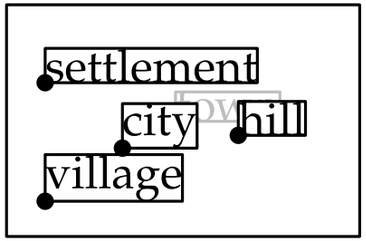


1S 2S 4S

Geometrische Beschriftungsmodelle

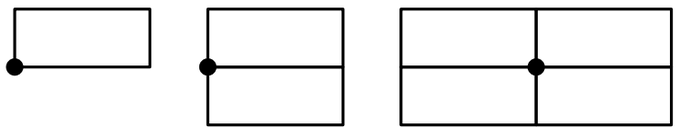
Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

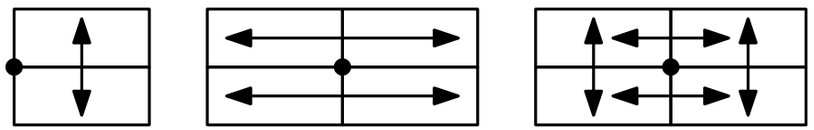


Was ist eine zulässige Beschriftung?

diskrete Modelle



Slider-Modelle

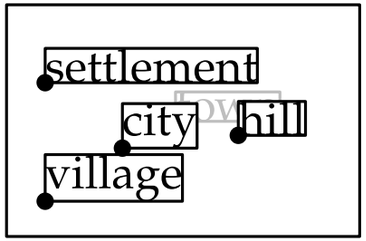


Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

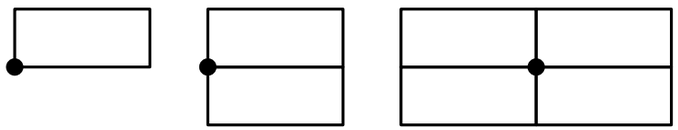
Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

oder zulässige Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)

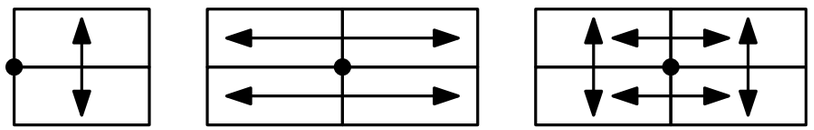


Was ist eine zulässige Beschriftung?

diskrete Modelle



Slider-Modelle

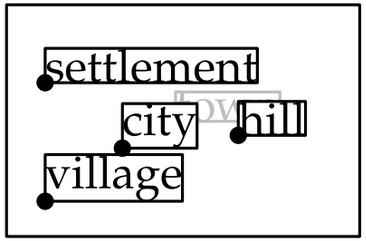


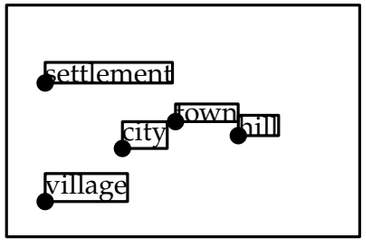
Geometrische Beschriftungsmodelle

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

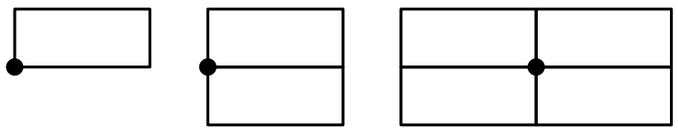
oder zulässige Beschriftung **aller** Label, so dass sich keine zwei Label schneiden und die **Schriftgröße** maximal ist (MaxSize)



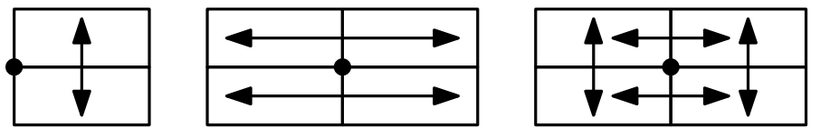


Was ist eine zulässige Beschriftung?

diskrete Modelle



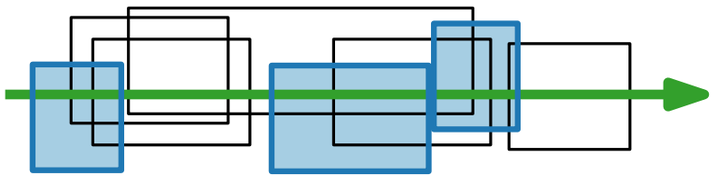
Slider-Modelle



Universität Trier

Algorithmen für geographische Informationssysteme

6. Vorlesung
Kartenbeschriftung



Teil II:
Diskretes Modell

Philipp Kindermann

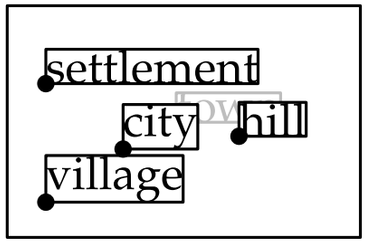
Wintersemester 2024/25

Umformulierung

MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

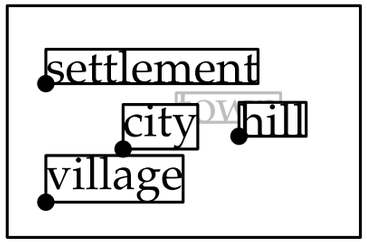


Umformulierung

MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



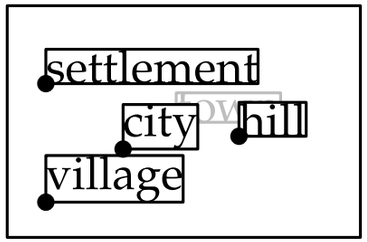
Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

Umformulierung

MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



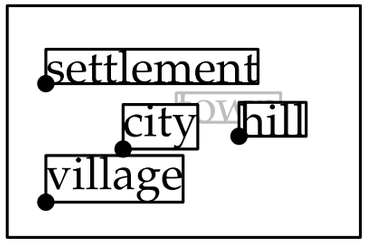
Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

Umformulierung

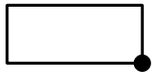
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

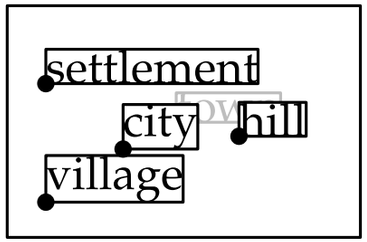


Umformulierung

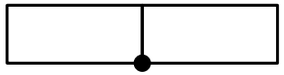
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

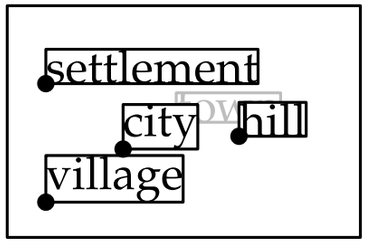


Umformulierung

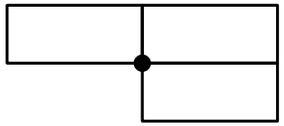
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

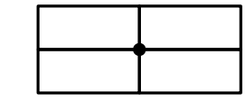
Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

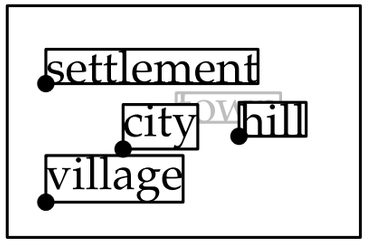


Umformulierung

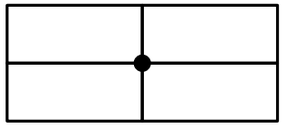


Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

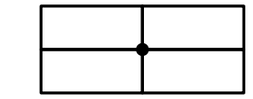
Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

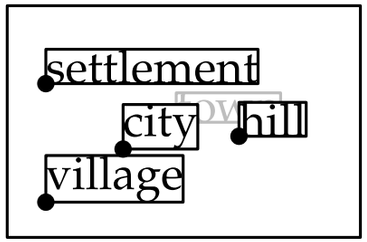


Umformulierung

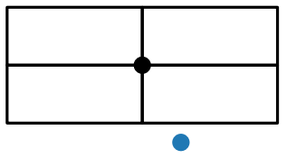


Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

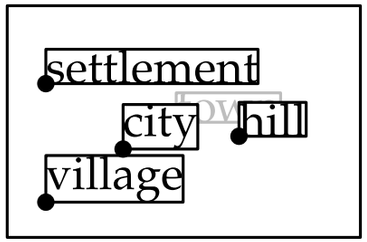


Umformulierung

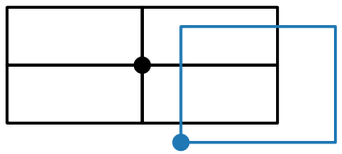
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

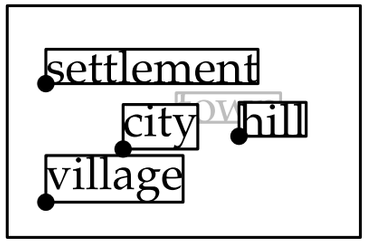


Umformulierung

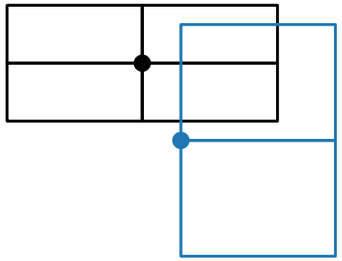
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

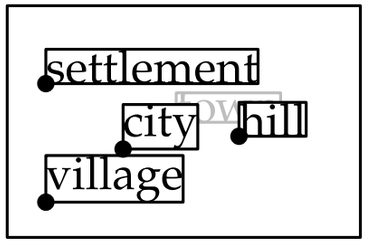


Umformulierung

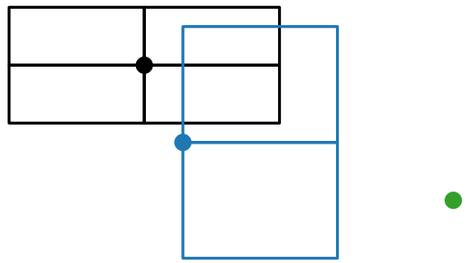
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

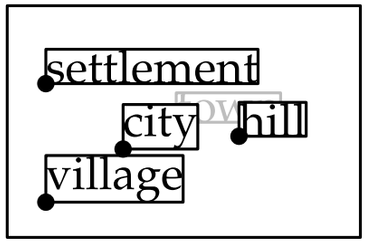


Umformulierung

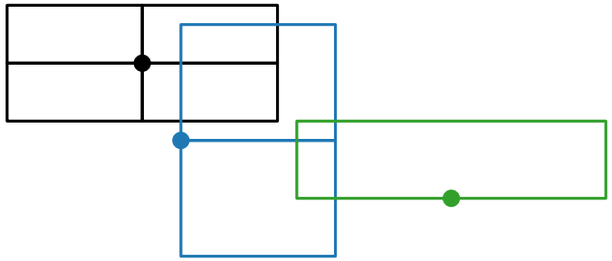
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

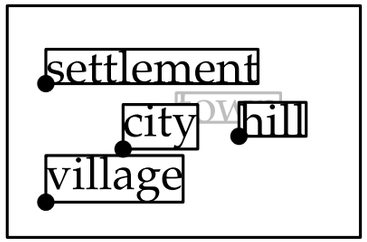


Umformulierung

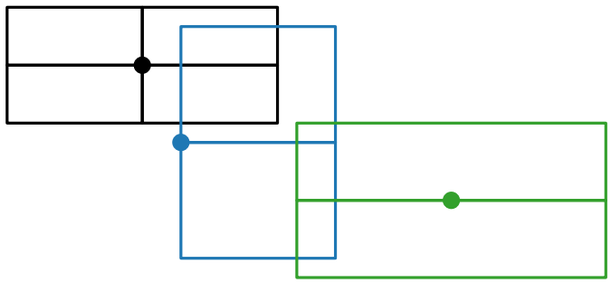
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

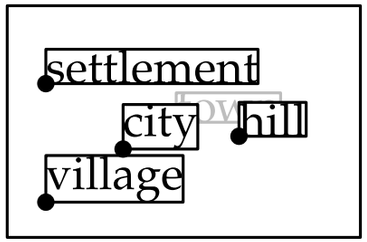


Umformulierung

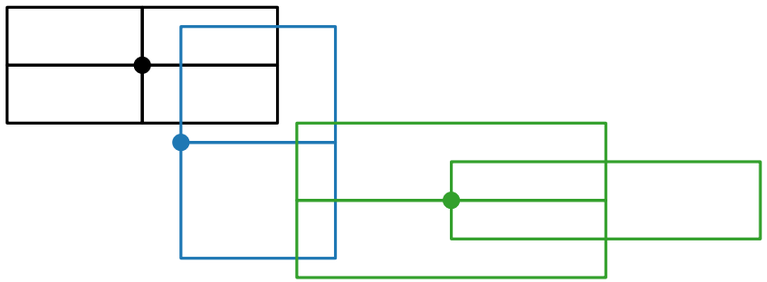
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

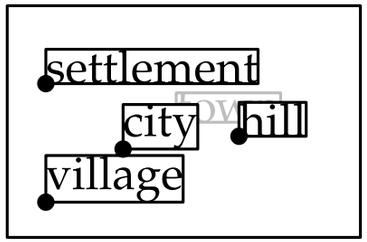


Umformulierung

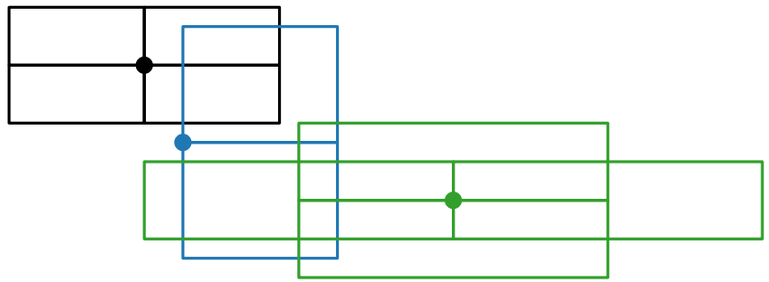
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

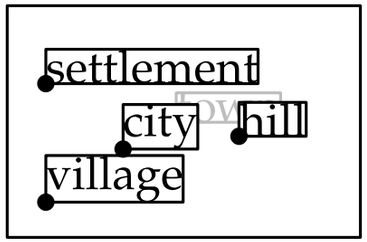


Umformulierung

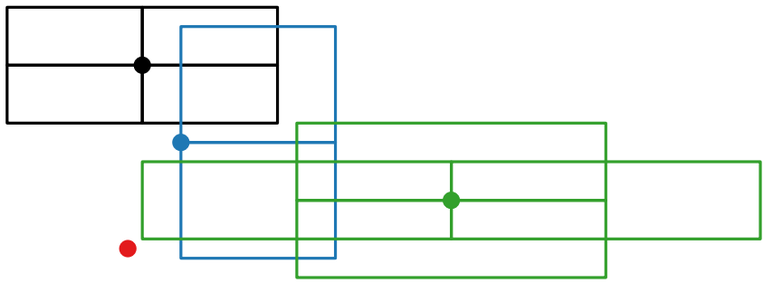
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

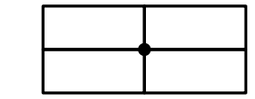
Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

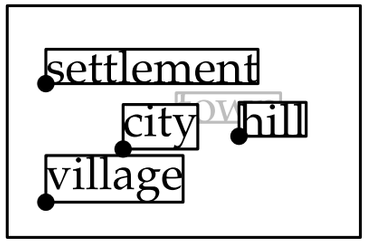


Umformulierung

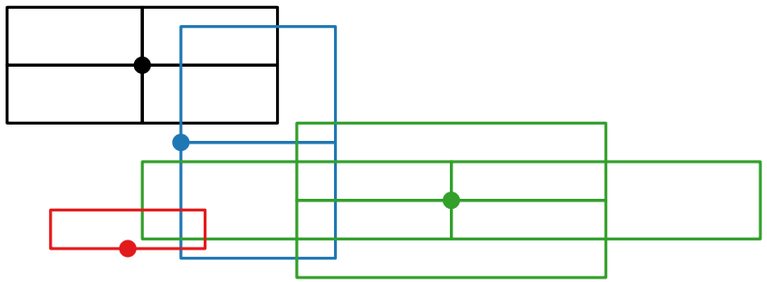


Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

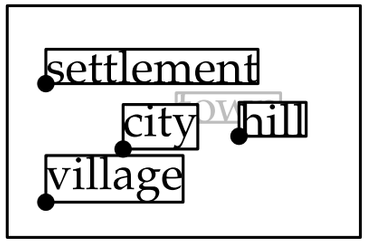


Umformulierung

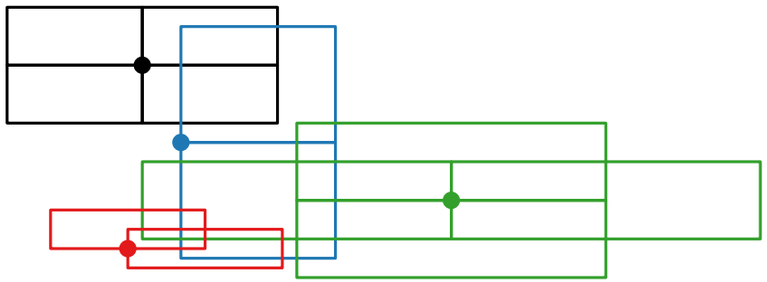
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

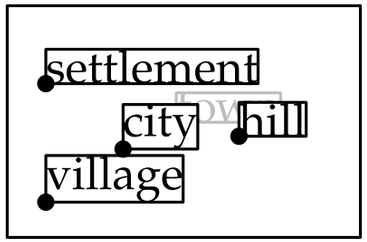


Umformulierung

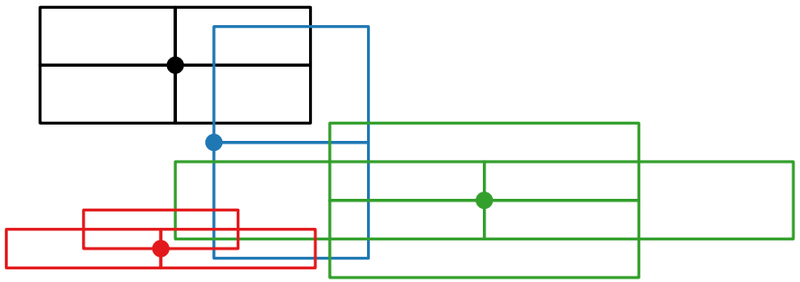
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.

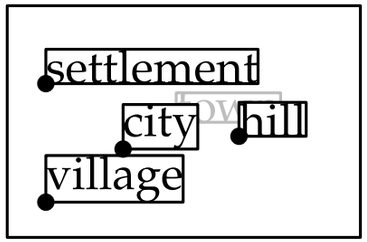


Umformulierung

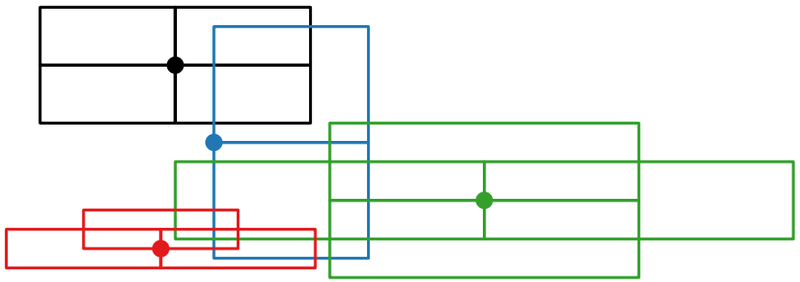
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



Geg.:

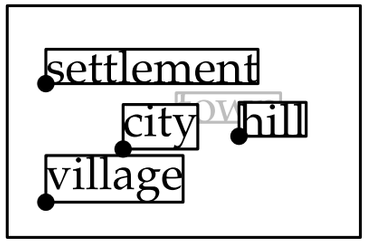
Ges.:

Umformulierung

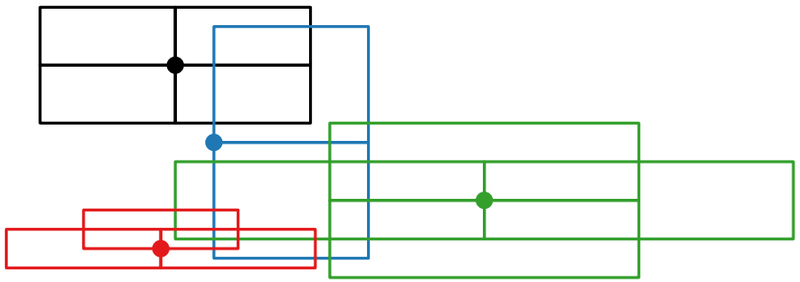
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



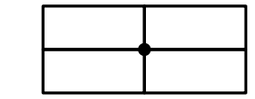
Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

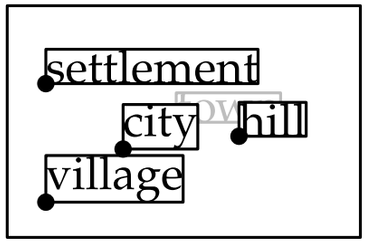
Ges.:

Umformulierung

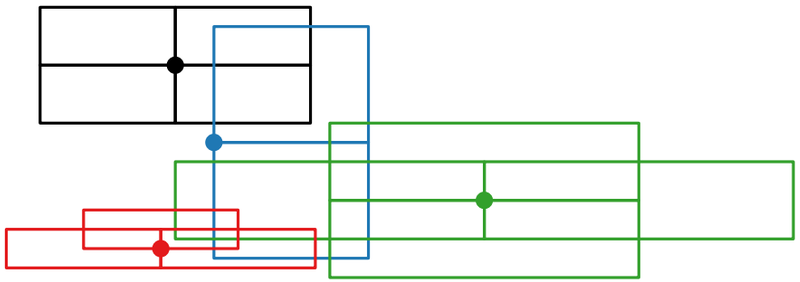


Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

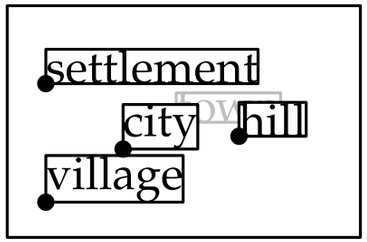
Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Umformulierung

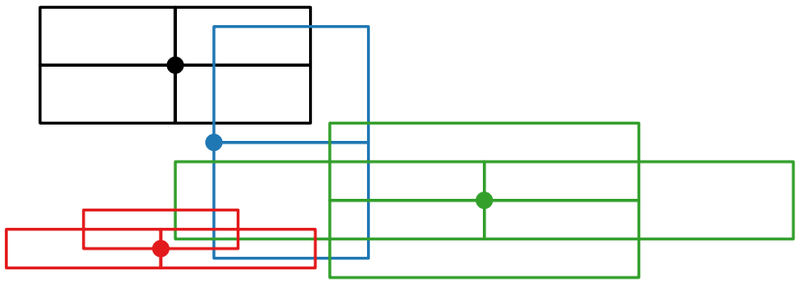
MaxNumber 4P

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)



Für jeden Punkt $p_i, 1 \leq i \leq m$ gibt es eine diskrete Menge von Labelpositionen, d.h., eine Menge R_i von achsenparallelen Rechtecken, wobei $p_i \in r$ für jedes $r \in R_i$.



Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

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

Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

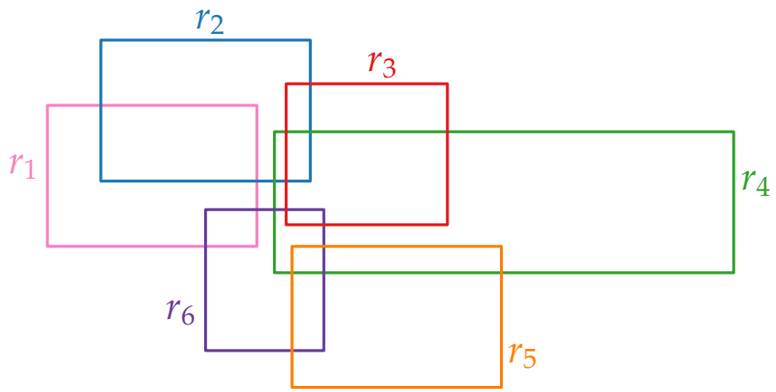
Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

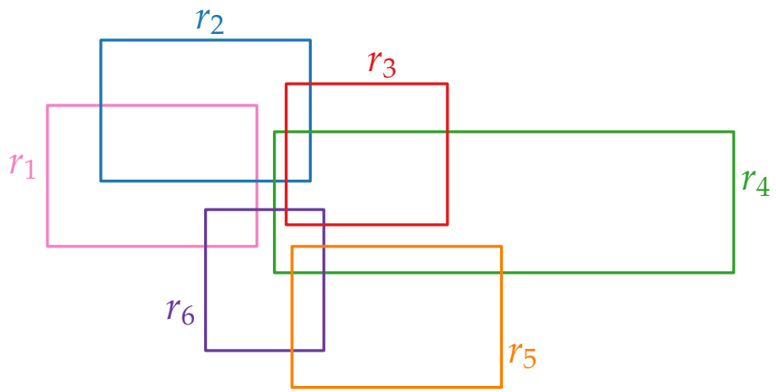


Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung



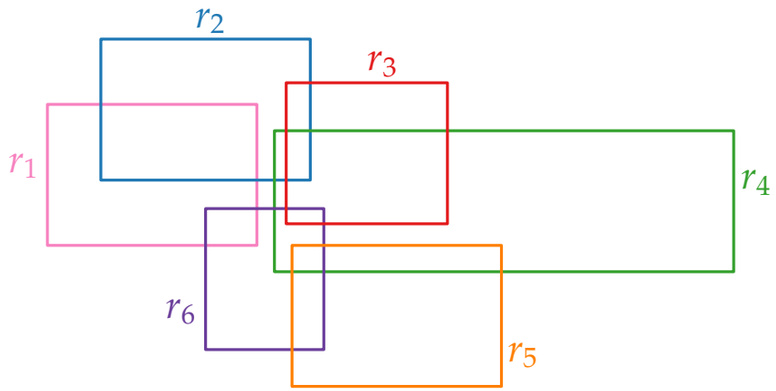
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

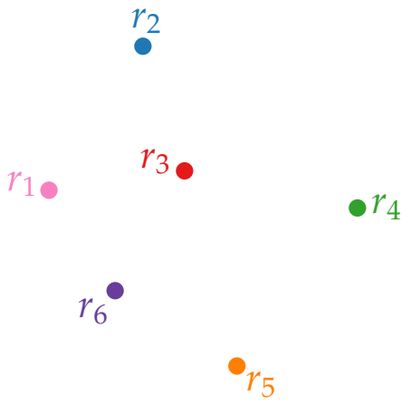
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





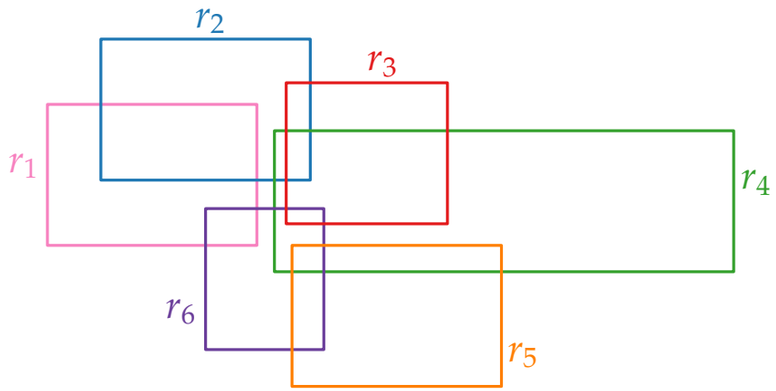
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

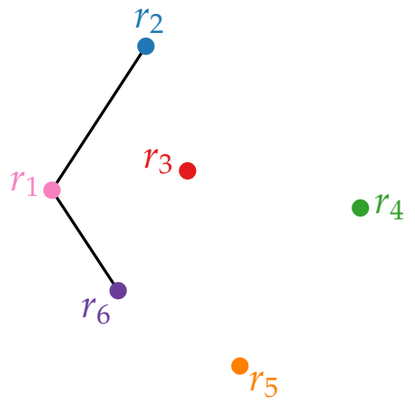
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





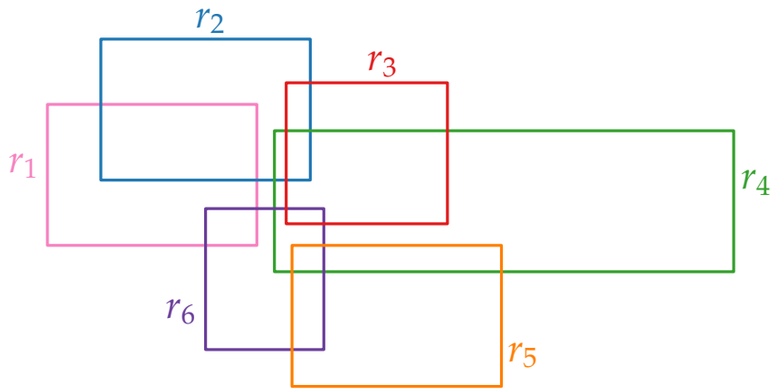
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

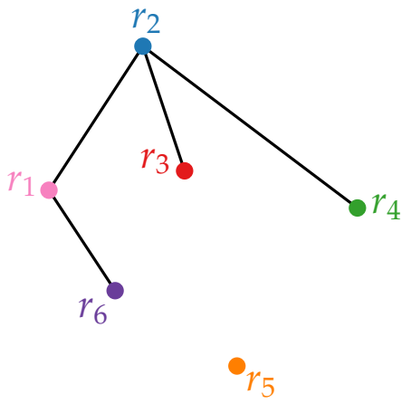
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





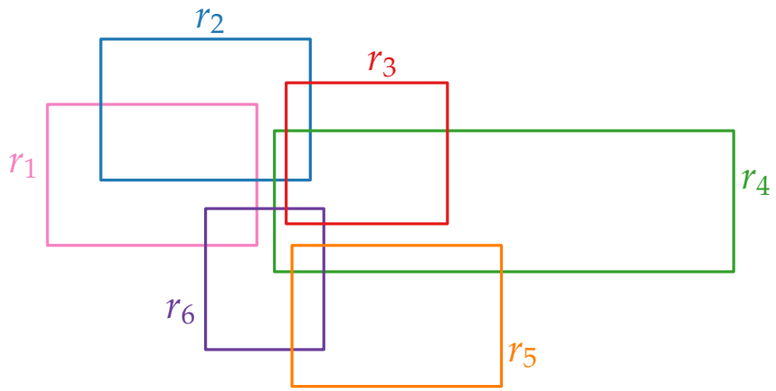
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

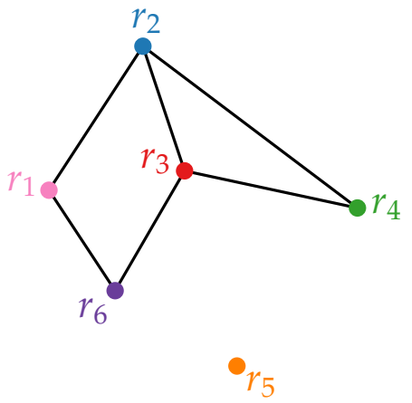
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





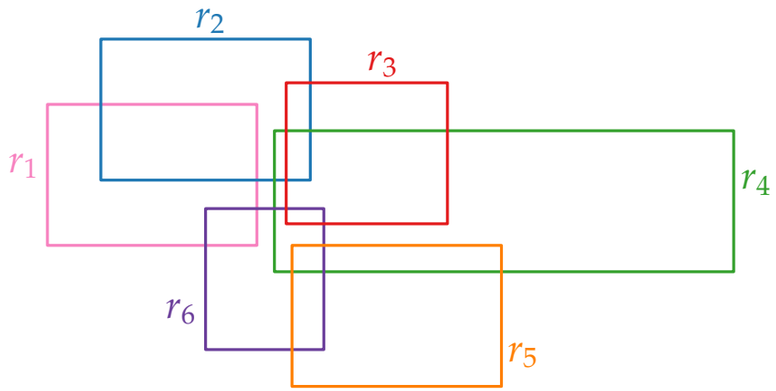
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

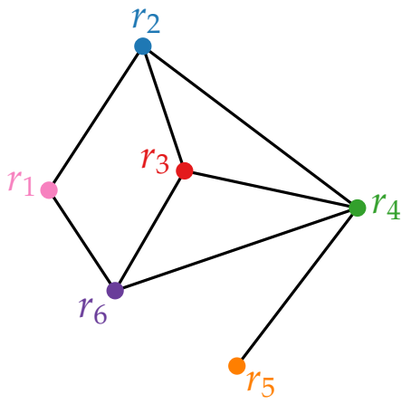
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





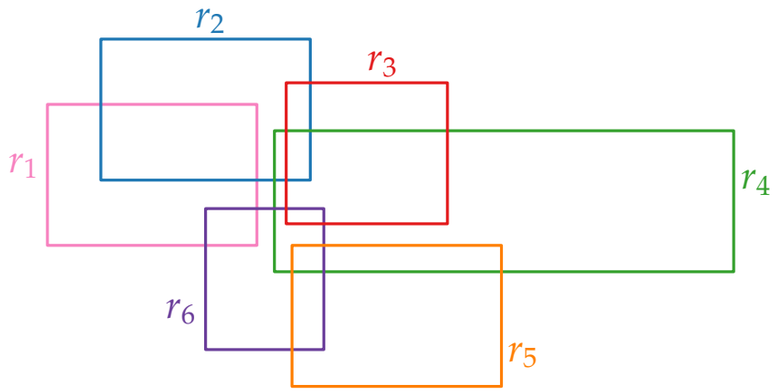
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

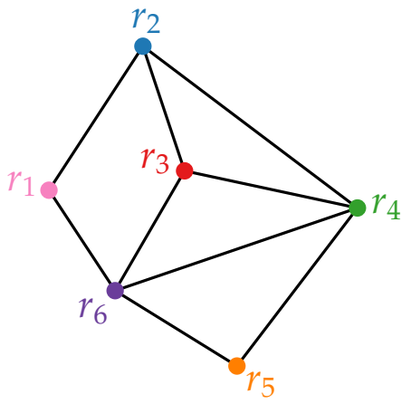
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





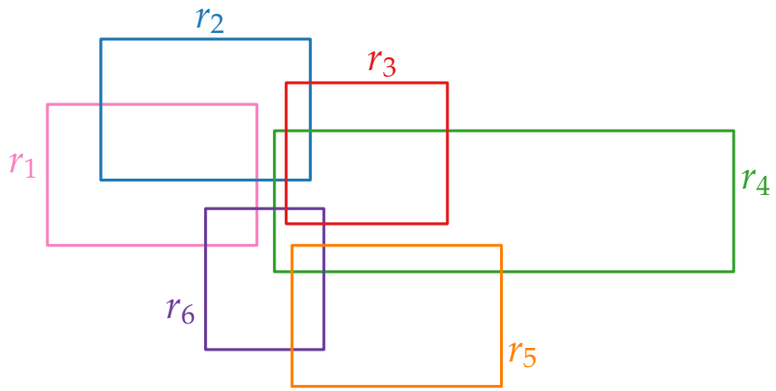
Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

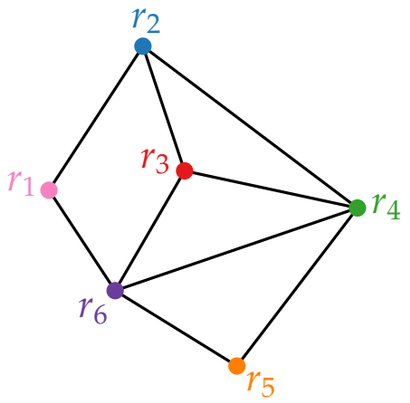
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

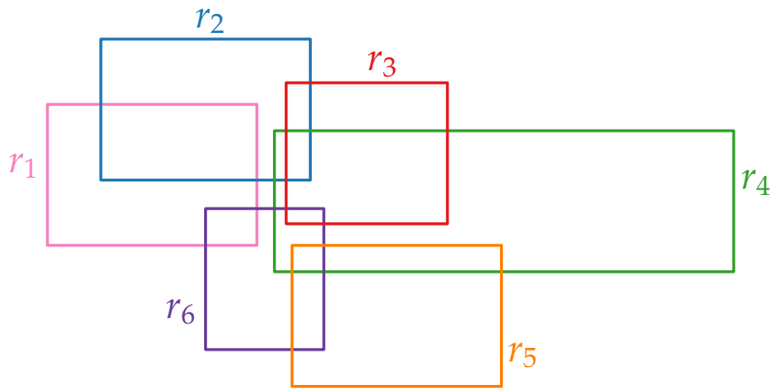
Lösung:

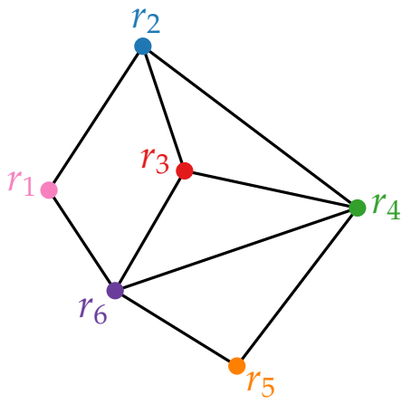
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

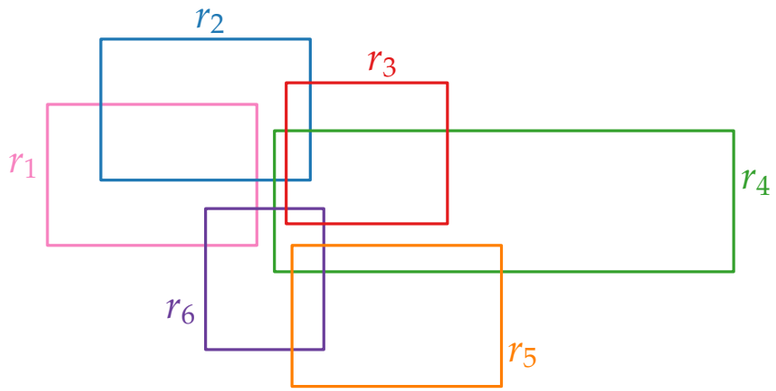
Lösung: Finde **größte unabhängige Menge** in G

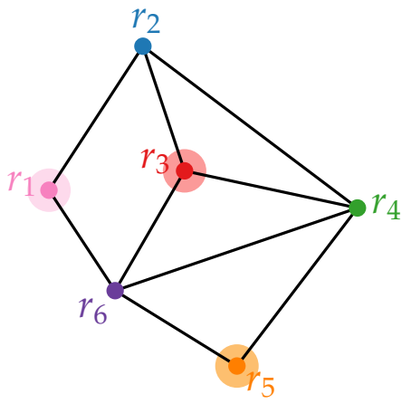
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

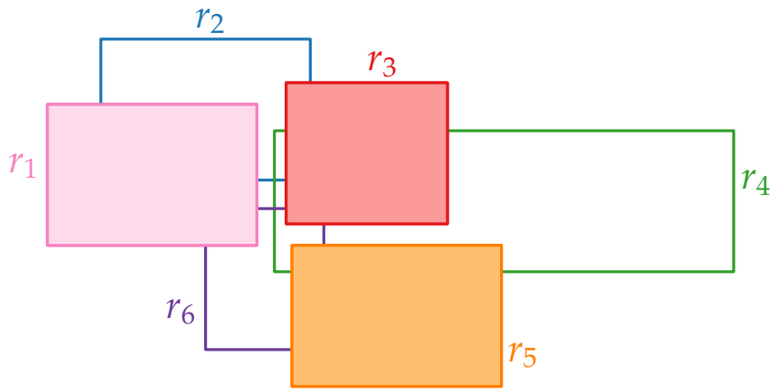
Lösung: Finde **größte unabhängige Menge** in G

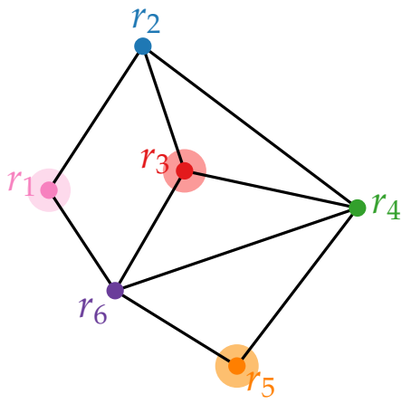
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

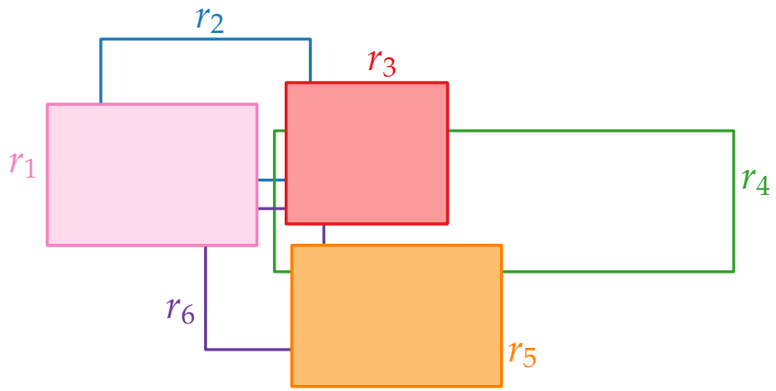
Lösung: Finde **größte unabhängige Menge** in G

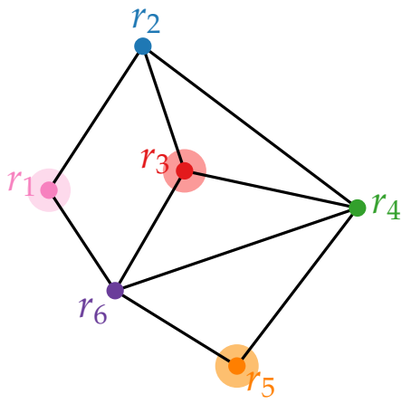
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

Lösung: Finde **größte unabhängige Menge** in G

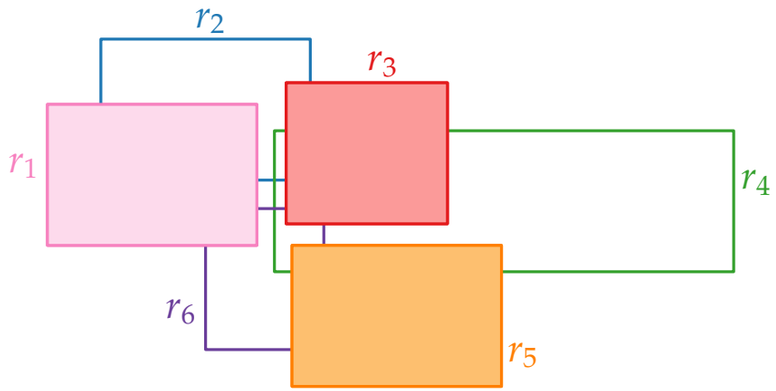
- Für allgemeine Graphen NP-schwer

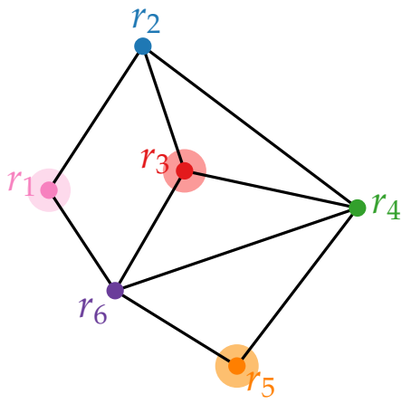
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

Lösung: Finde **größte unabhängige Menge** in G

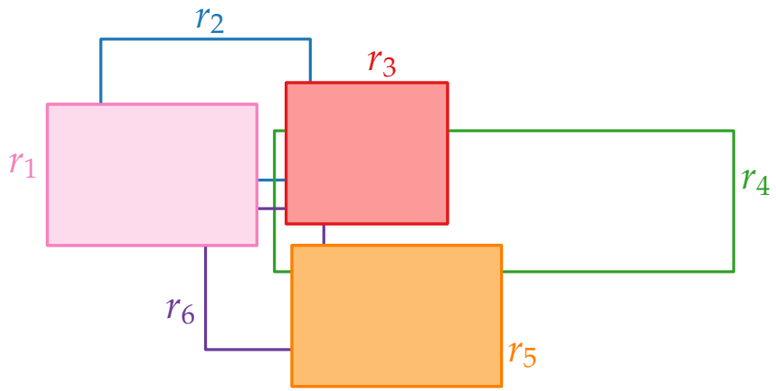
- Für allgemeine Graphen NP-schwer
- Für Schnittgarphen von uniformen Quadraten NP-schwer [Imai & Asano 1983]

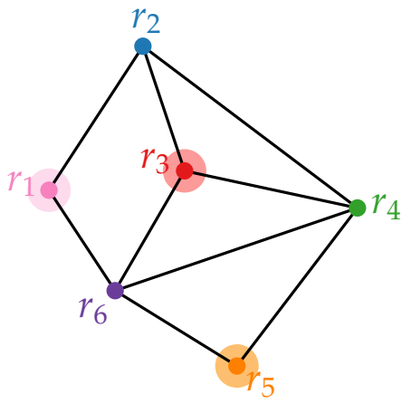
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

Lösung: Finde **größte unabhängige Menge** in G

- Für allgemeine Graphen NP-schwer
- Für Schnittgarphen von uniformen Quadraten NP-schwer [Imai & Asano 1983]

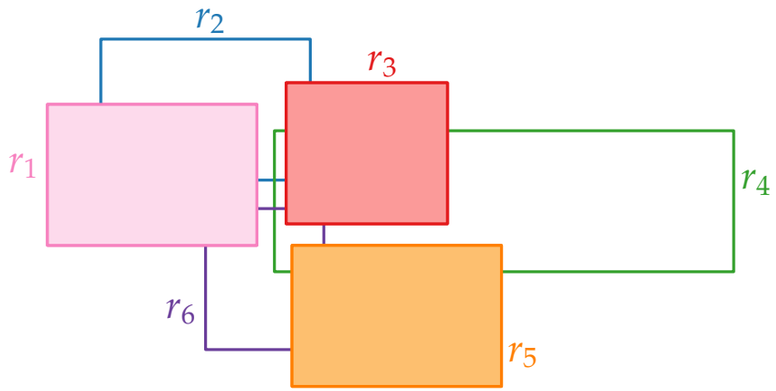
Ziel:

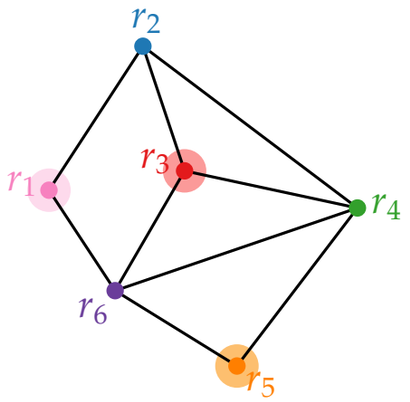
Schnittgraph

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung





Der **Schnittgraph** $G = (R, E)$ enthält einen Knoten für jedes Rechteck und eine Kante (u, v) genau dann wenn sich die Rechtecke u und v schneiden.

Lösung: Finde **größte unabhängige Menge** in G

- Für allgemeine Graphen NP-schwer
- Für Schnittgarphen von uniformen Quadraten NP-schwer [Imai & Asano 1983]

Ziel: Approximationsalgorithmus

Spezialfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Spezialfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

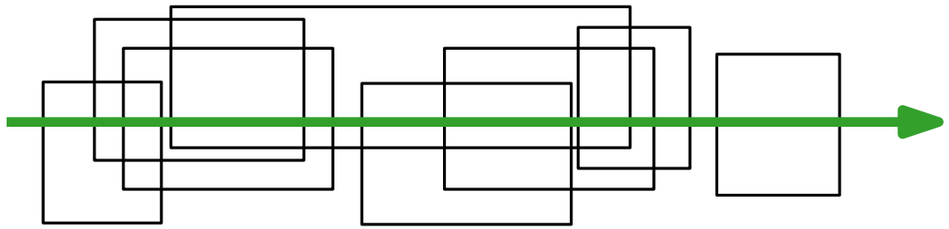
Spezialfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.



Spezialfall

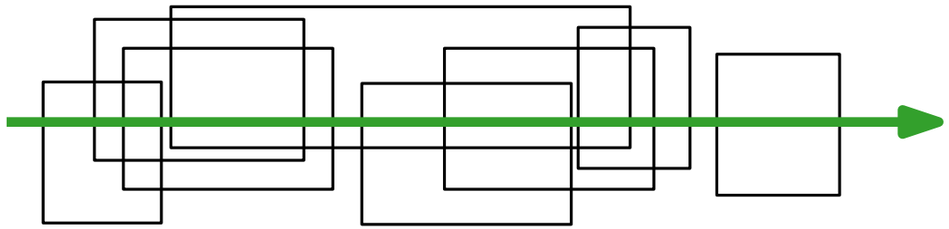
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie:



Spezialfall

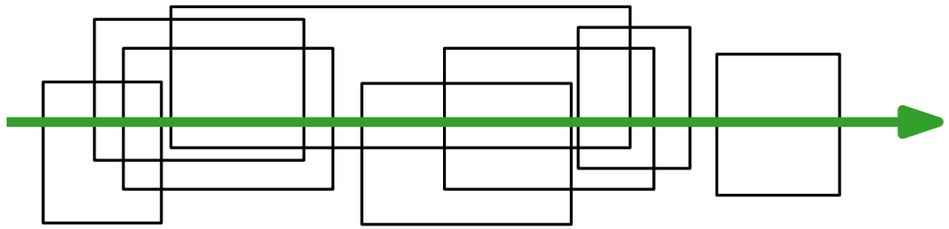
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



Spezialfall

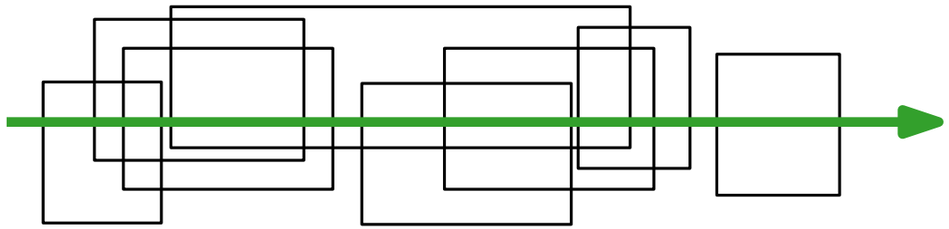
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes

Spezialfall

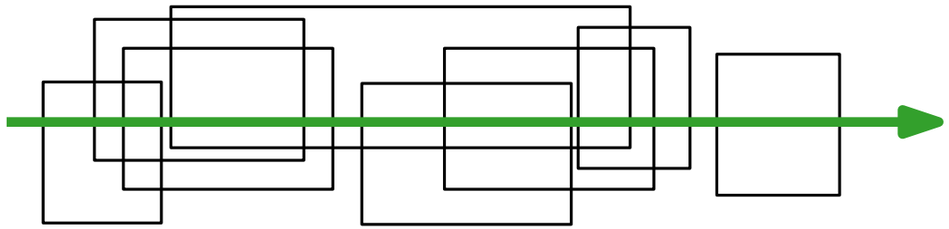
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head

Spezialfall

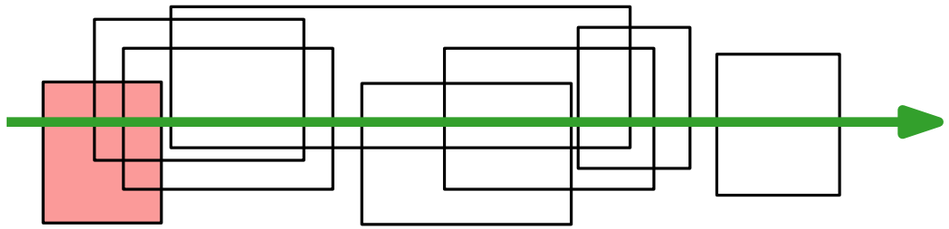
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head

Spezialfall

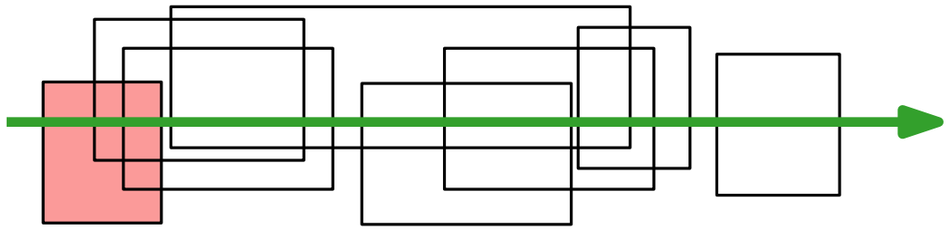
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head

return S

Spezialfall

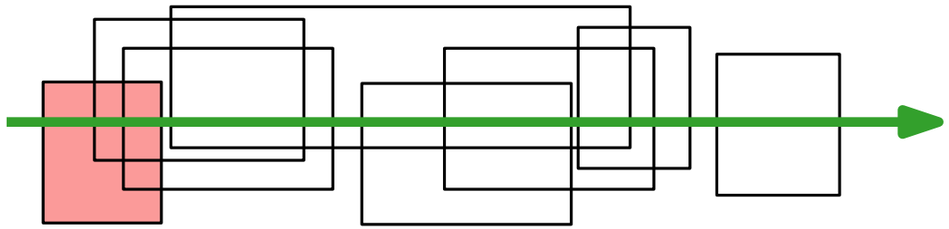
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**

return S

Spezialfall

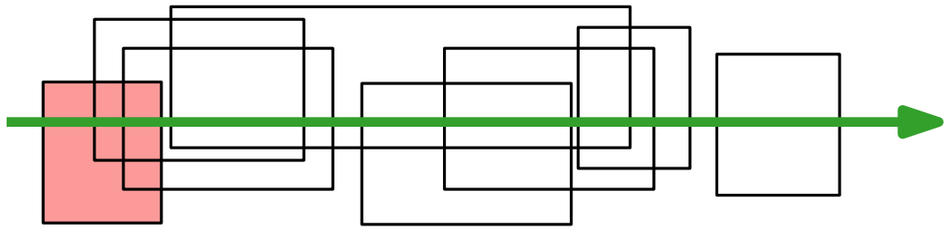
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
    $S = S \cup \{r\}$;
return S

Spezialfall

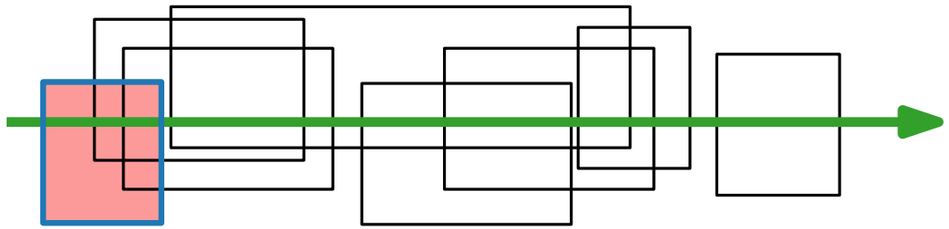
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
    $S = S \cup \{r\}$;
return S

Spezialfall

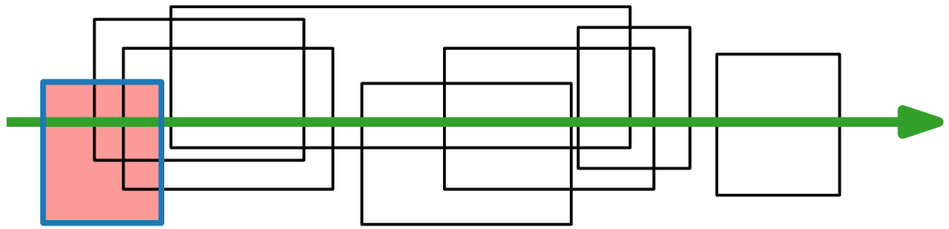
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
    $S = S \cup \{r\}$; $r' = r$.next
return S

Spezialfall

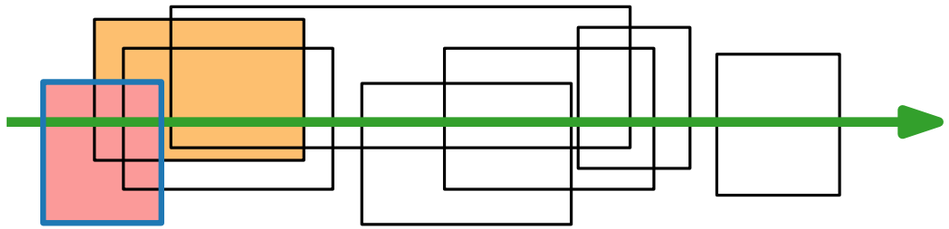
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
    $S = S \cup \{r\}$; $r' = r$.next
return S

Spezialfall

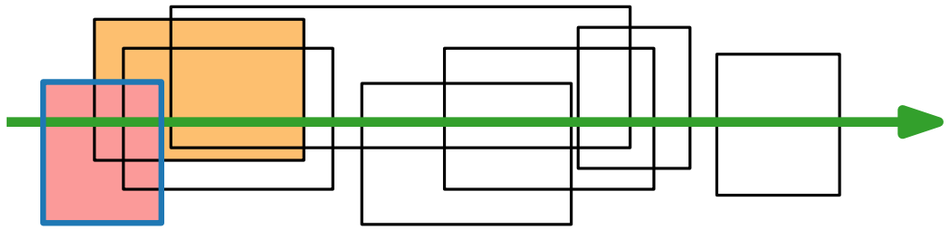
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
  $S = S \cup \{r\}$; $r' = r$.next
  while $r' \neq$ **nil**

return S

Spezialfall

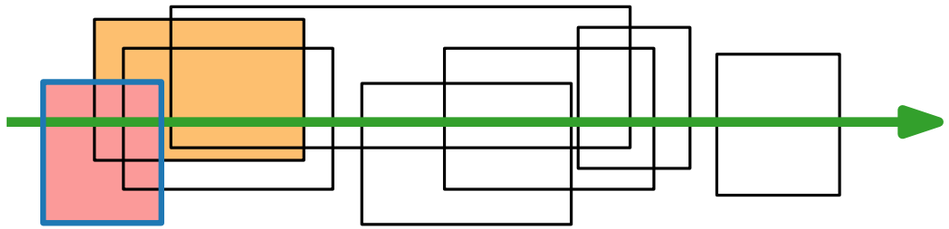
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.head
while $r \neq$ **nil do**
  $S = S \cup \{r\}$; $r' = r$.next
  while $r' \neq$ **nil and** r'.xMin $\leq r$.xMax **do**

return S

Spezialfall

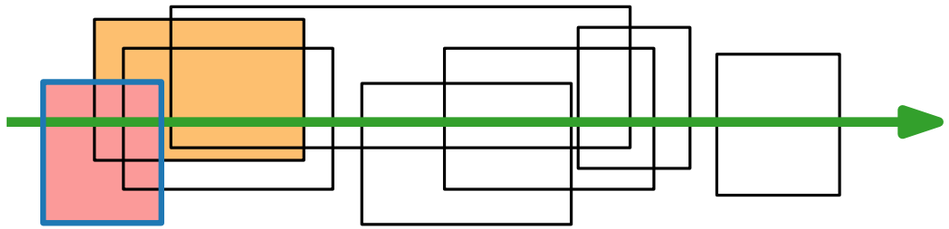
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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
return S

Spezialfall

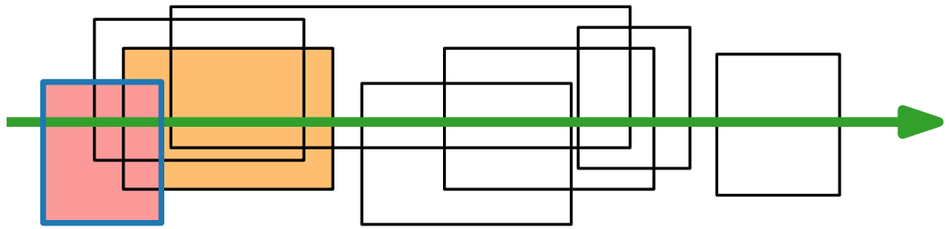
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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
return S

Spezialfall

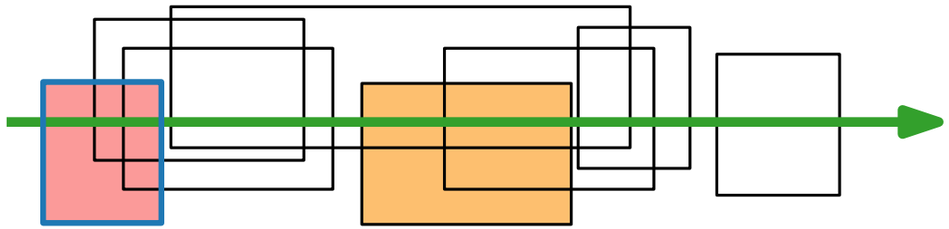
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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
return S

Spezialfall

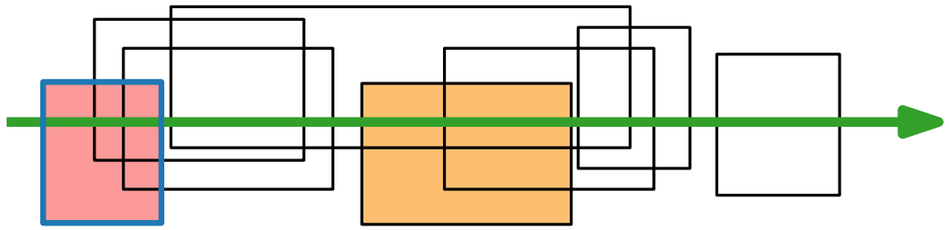
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

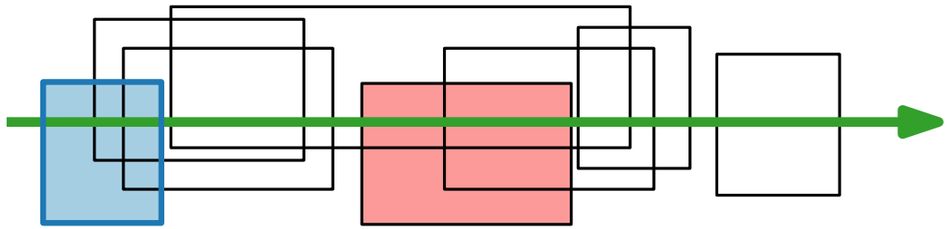
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

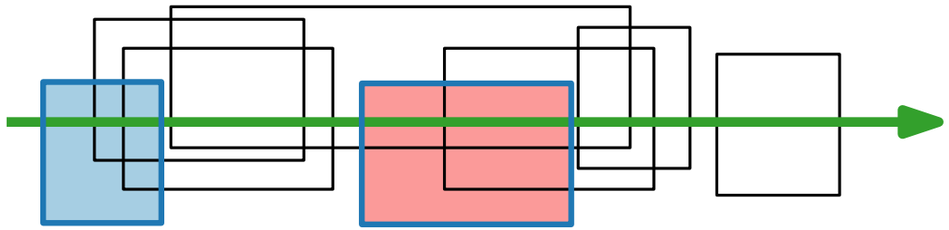
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

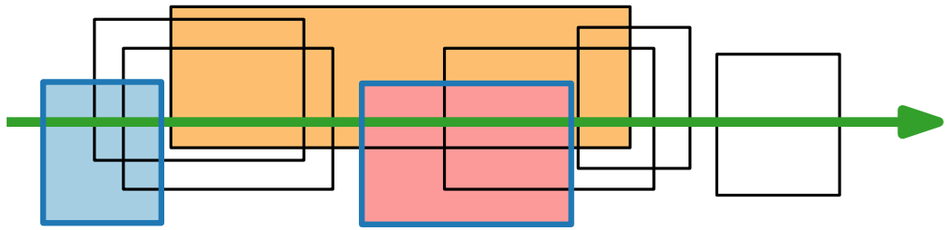
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

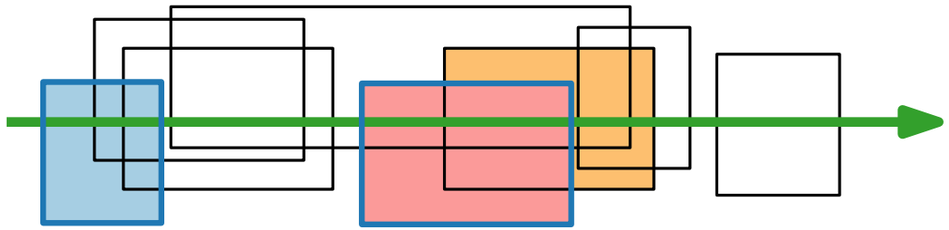
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

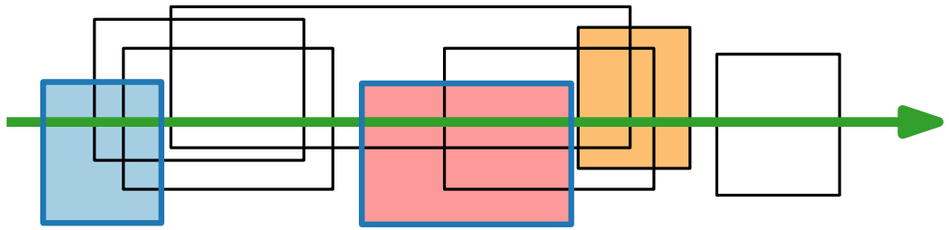
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

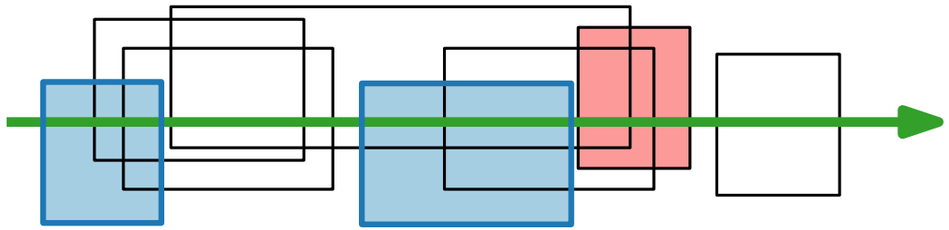
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

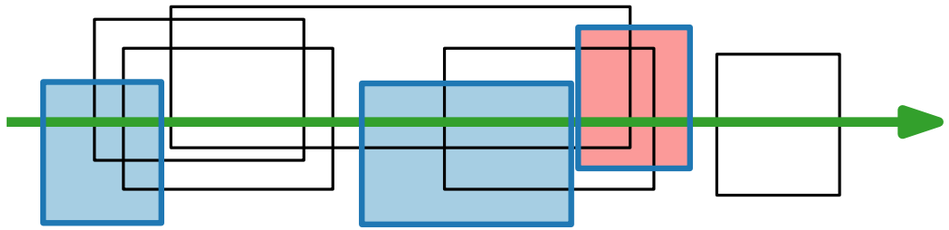
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

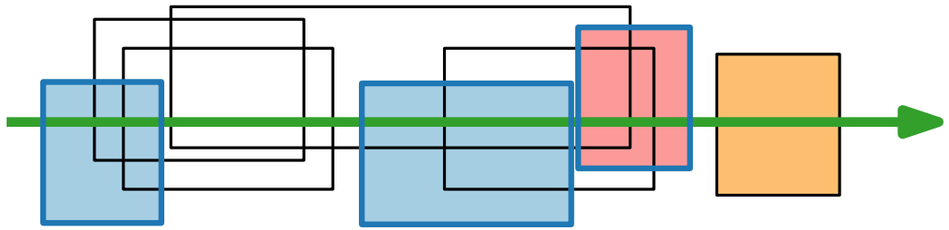
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

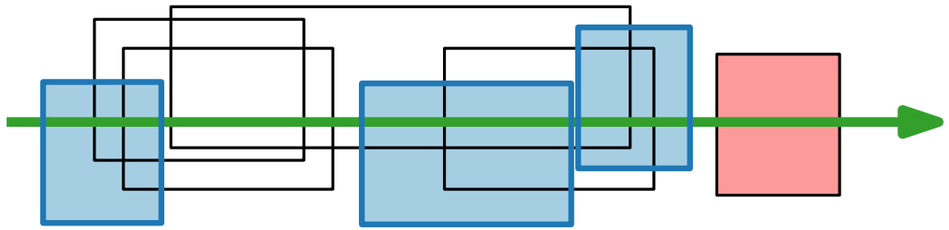
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

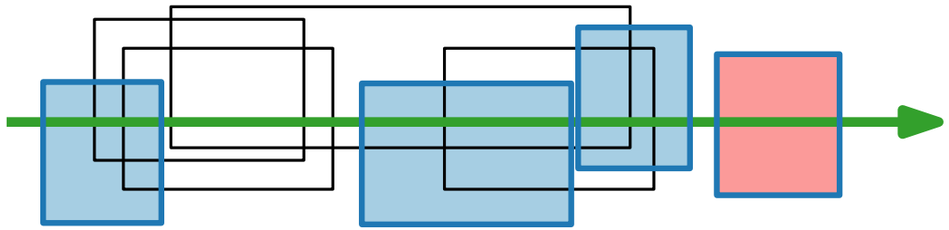
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

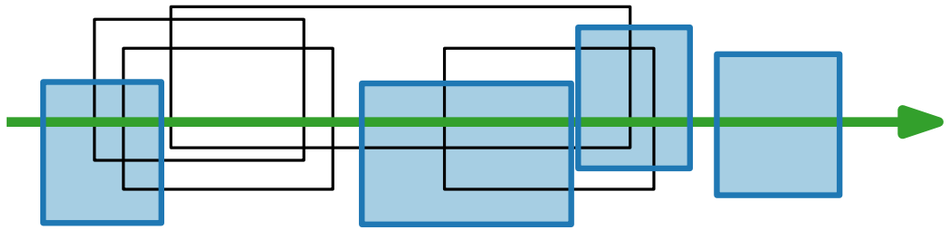
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Spezialfall

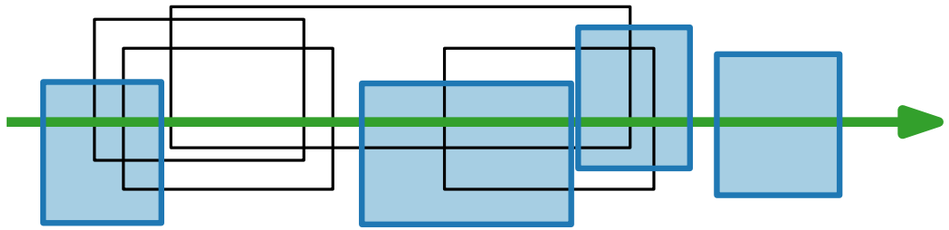
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Laufzeit?

Spezialfall

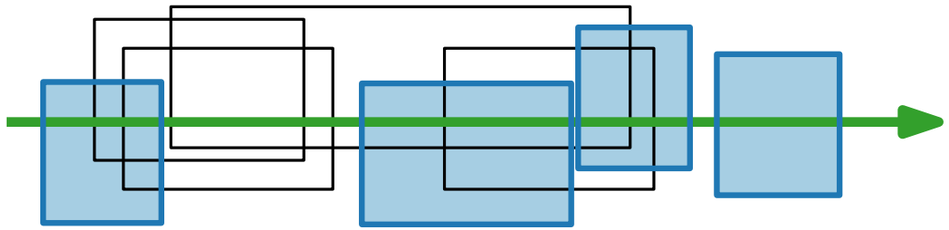
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Laufzeit?

- $\mathcal{O}(n \log n)$

Spezialfall

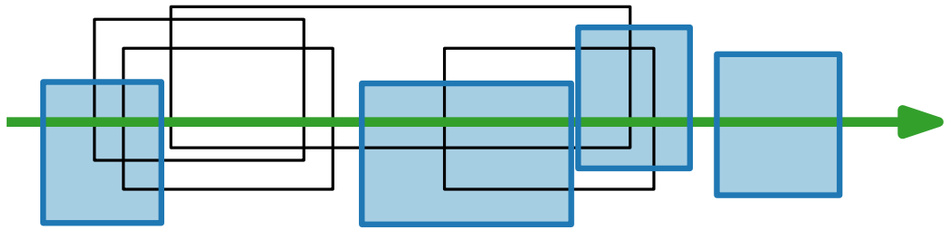
MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

Angenommen, es gibt eine horizontale Gerade, die alle Rechtecke in R schneidet.

Strategie: Greedy!



L = Rechtecke sortiert nach x Koordinate des rechten Randes
$S = \emptyset$; $r = L$.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

Laufzeit?

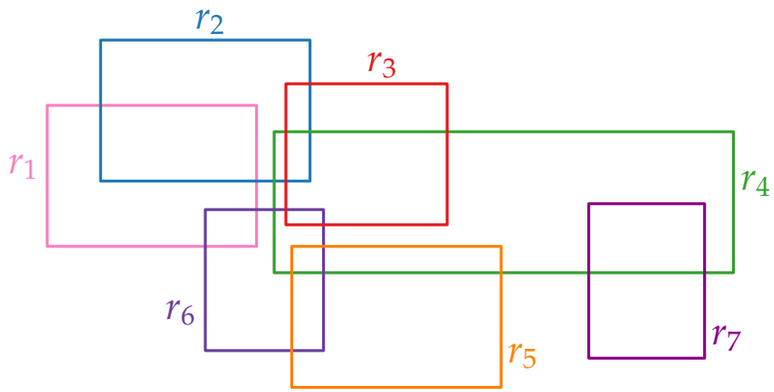
- $\mathcal{O}(n \log n)$
- $\mathcal{O}(n)$ falls vorsortiert

Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung



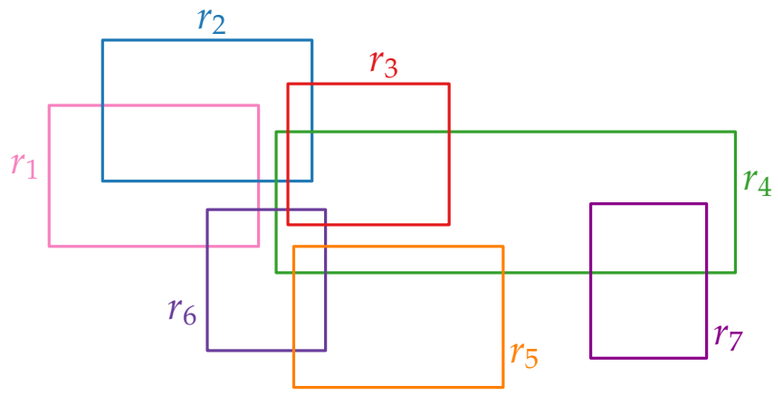
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$



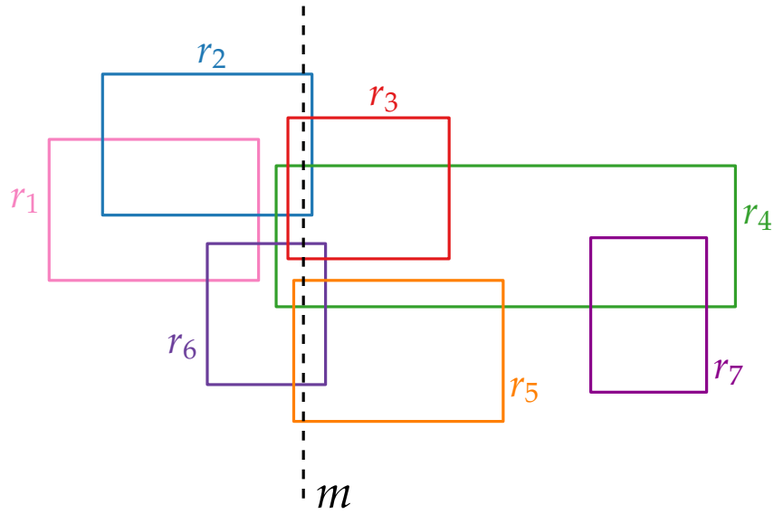
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$



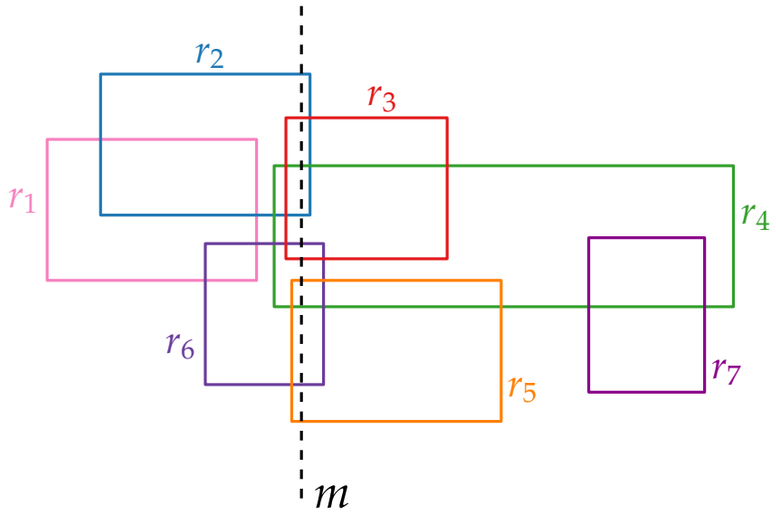
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m



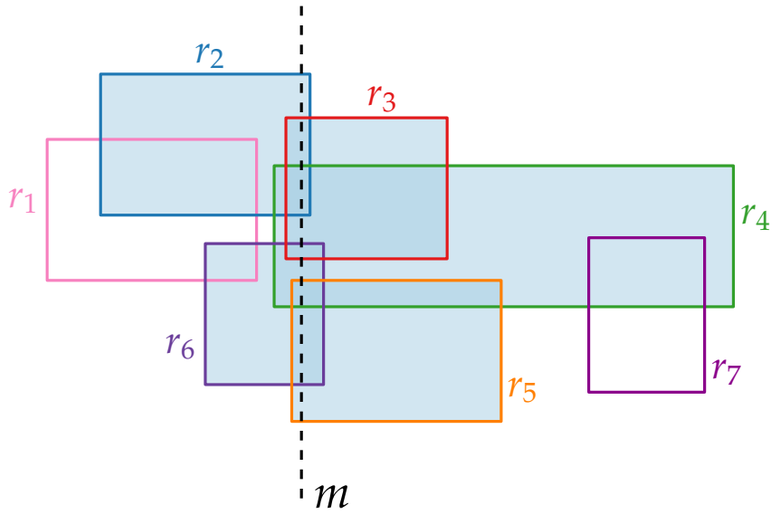
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m



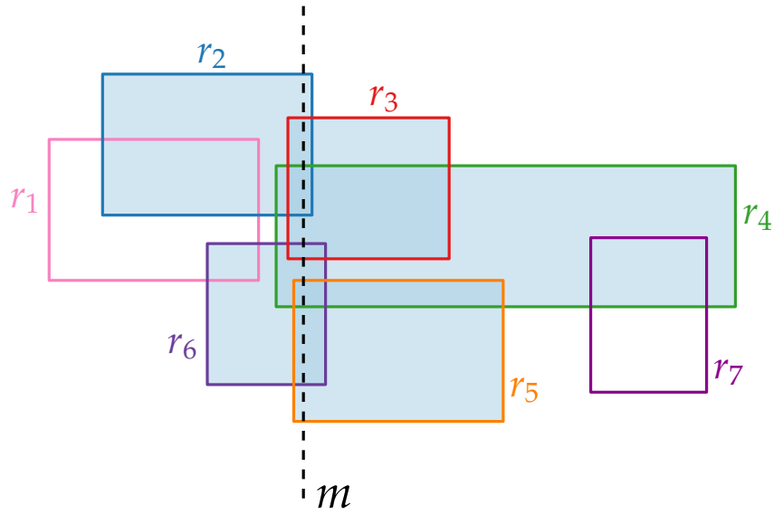
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m



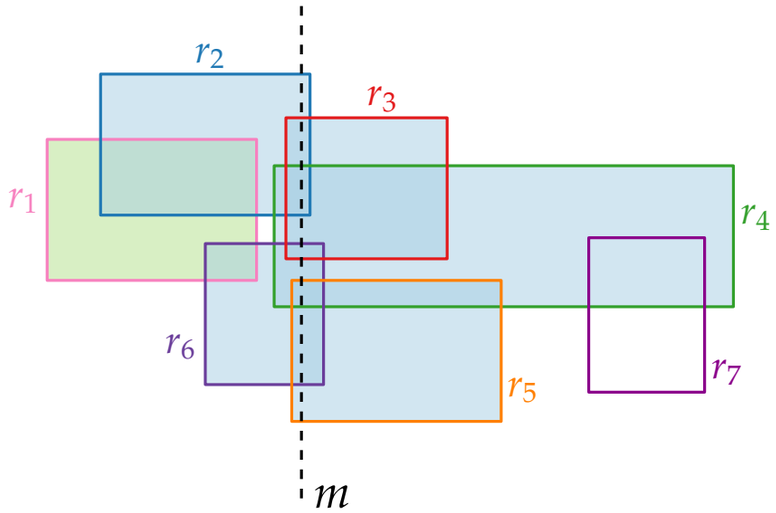
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m



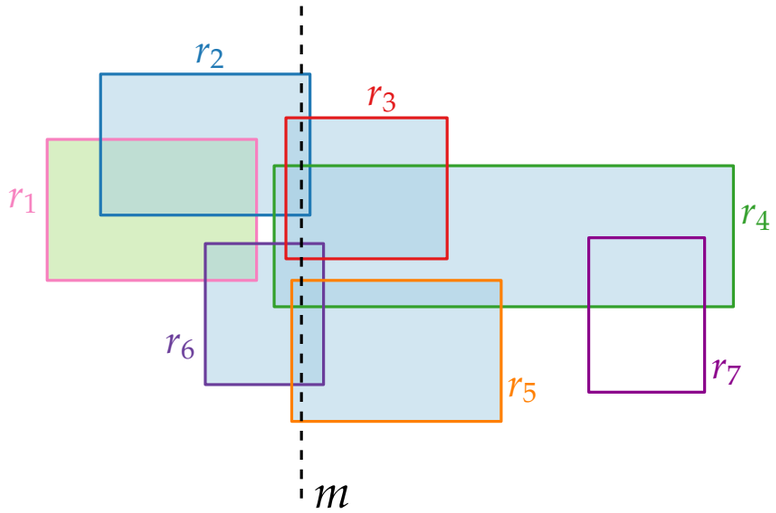
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



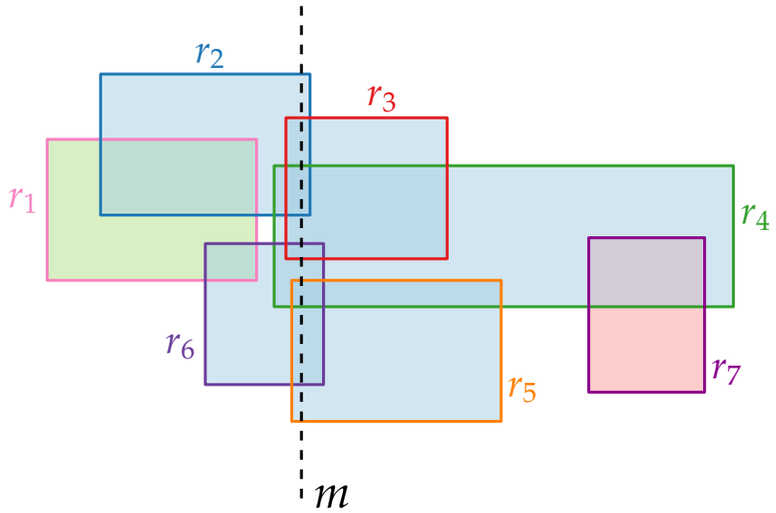
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



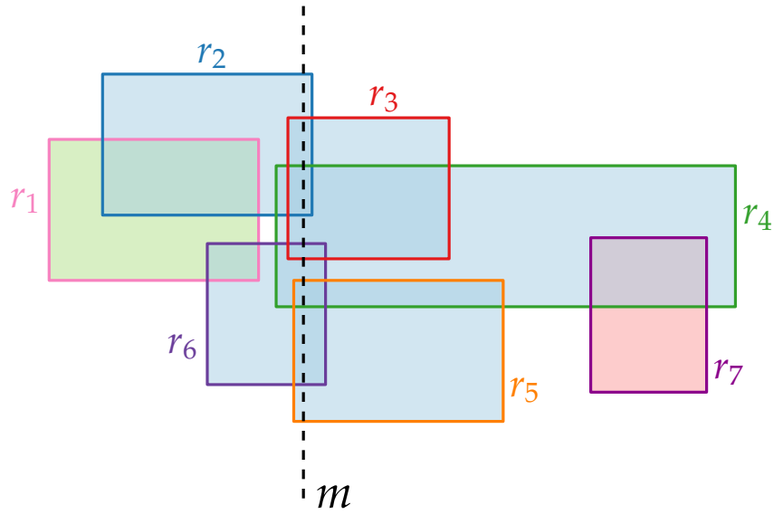
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

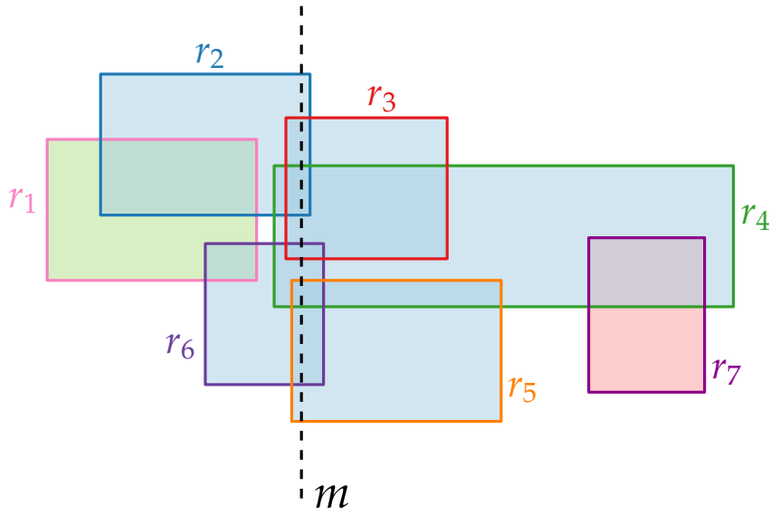
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

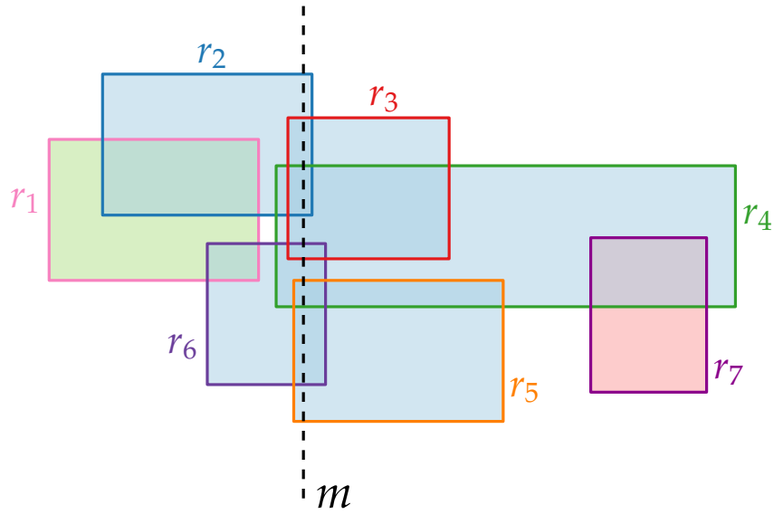
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

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

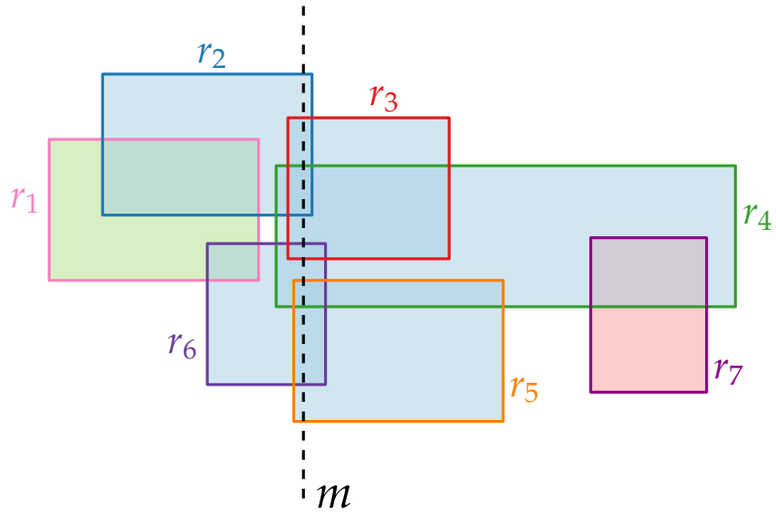
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$

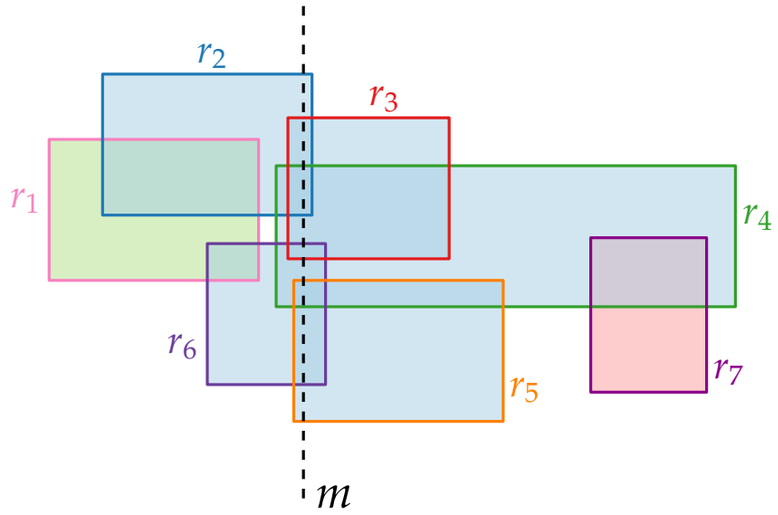
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $r_1 \cap r_2$ **then return** $\{r_1\}$

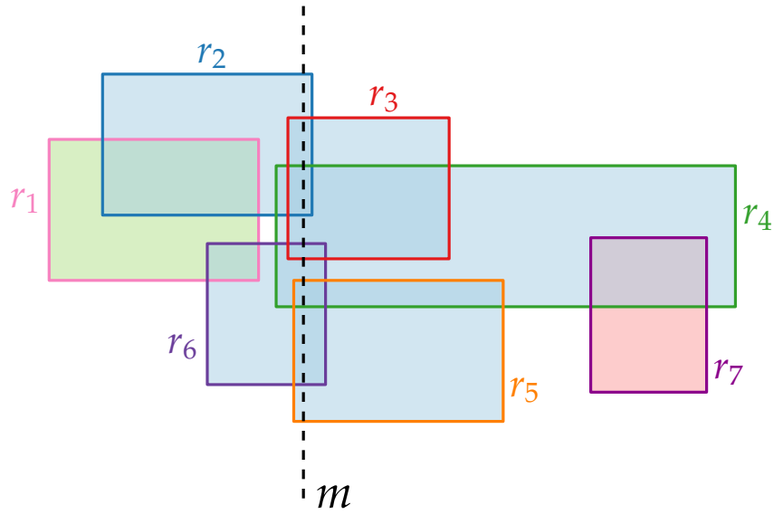
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



Algorithmus für $n \leq 2$:

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $r_1 \cap r_2$ **then return** $\{r_1\}$
return $\{r_1, r_2\}$

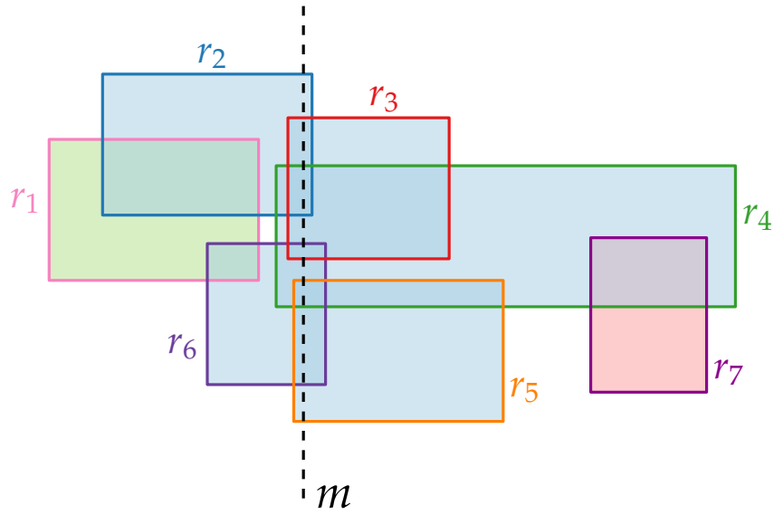
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

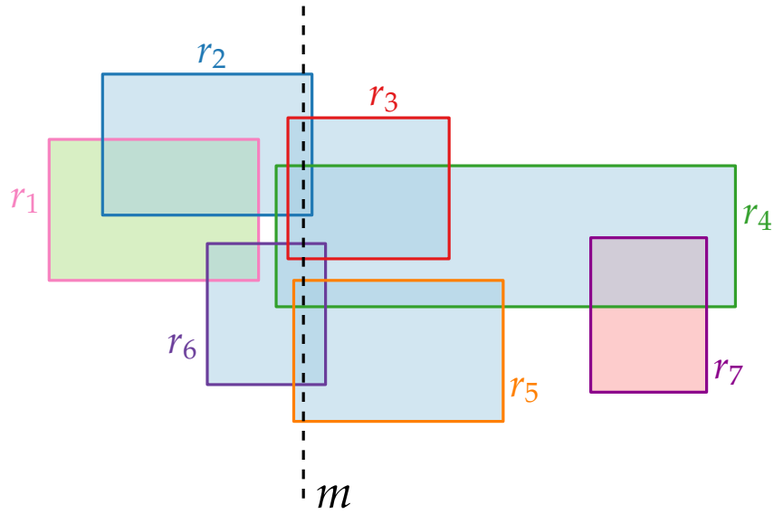
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r

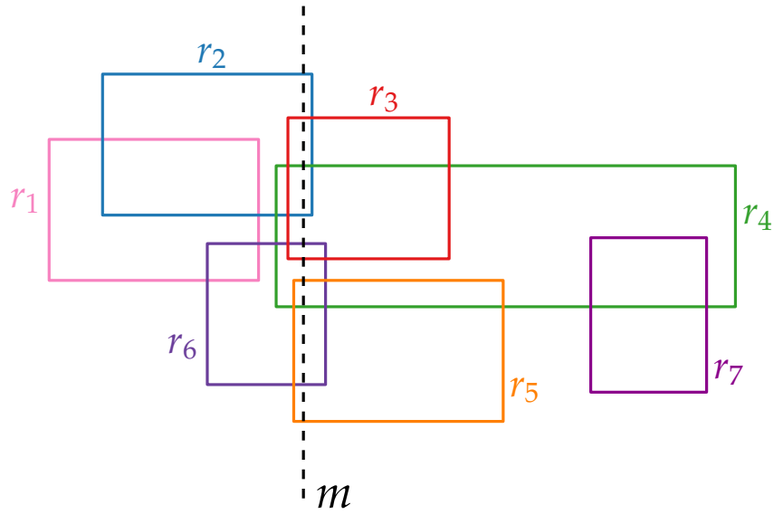
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r

L_m = GreedyMaxNumber(R_m)

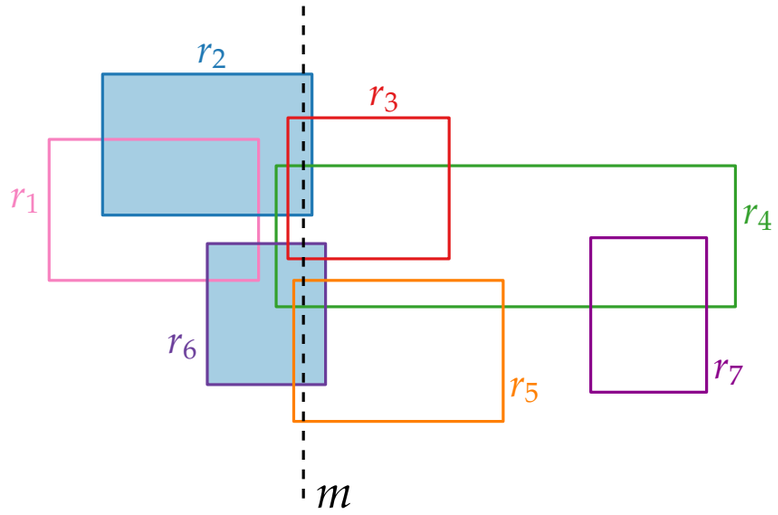
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r

L_m = GreedyMaxNumber(R_m)

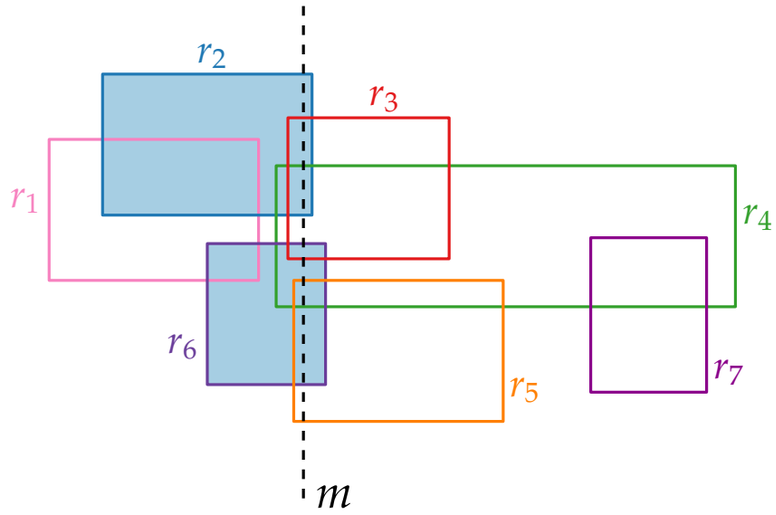
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

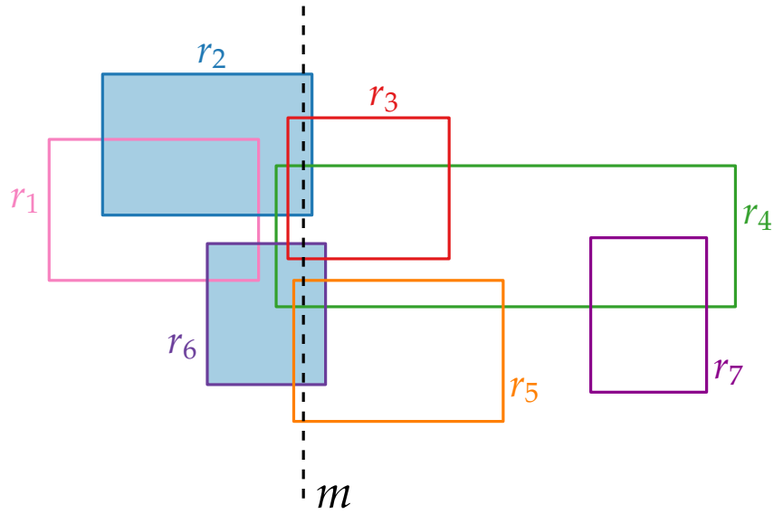
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

L_ℓ = MaxNumberRect(R_ℓ)

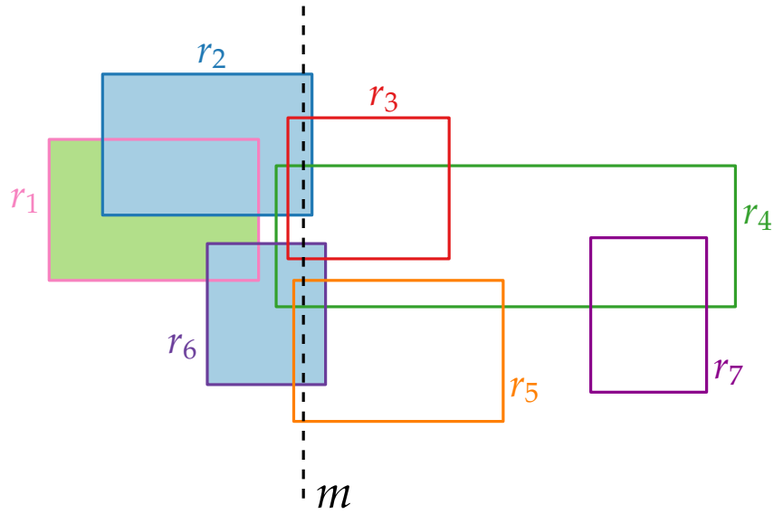
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

L_ℓ = MaxNumberRect(R_ℓ)

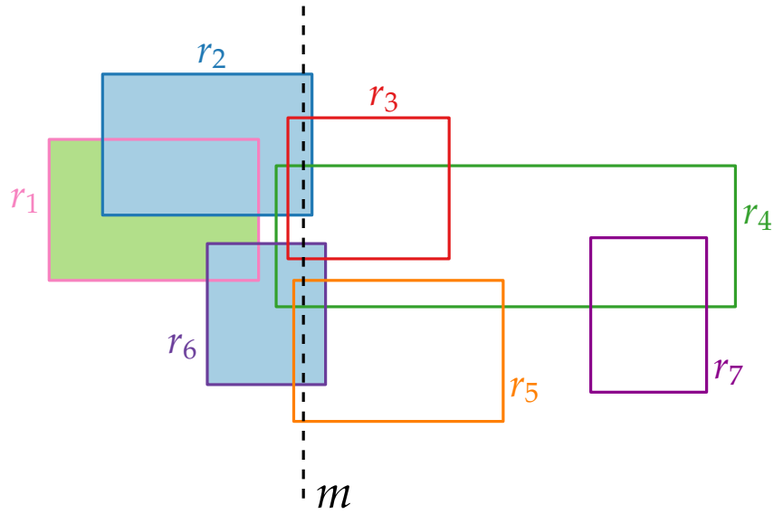
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

L_ℓ = MaxNumberRect(R_ℓ)

L_r = MaxNumberRect(R_r)

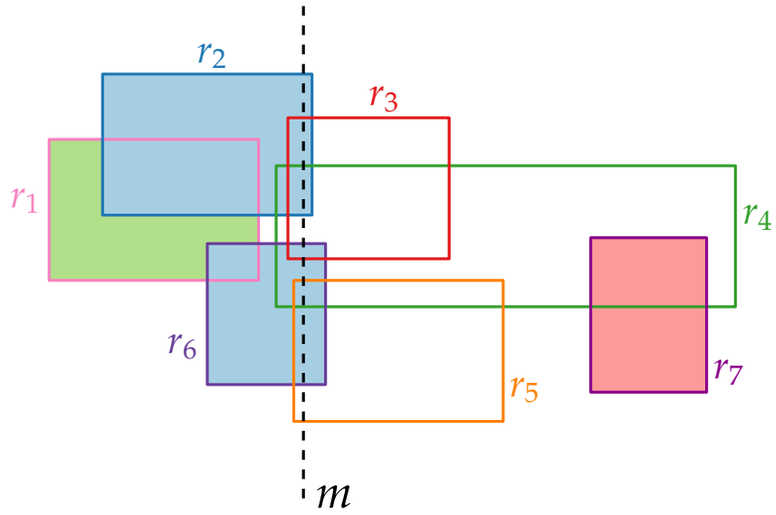
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

L_ℓ = MaxNumberRect(R_ℓ)

L_r = MaxNumberRect(R_r)

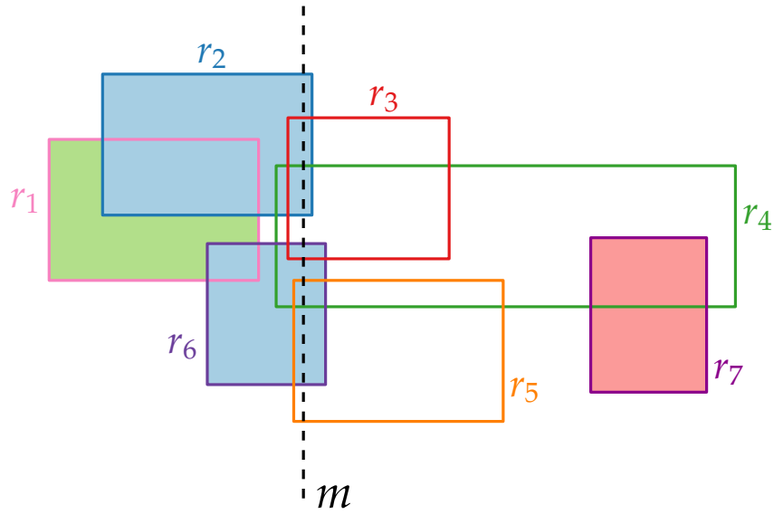
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)

L_m = GreedyMaxNumber(R_m)

L_ℓ = MaxNumberRect(R_ℓ)

L_r = MaxNumberRect(R_r)

if $|L_m| \geq |L_\ell| + |L_r|$ **then**

└

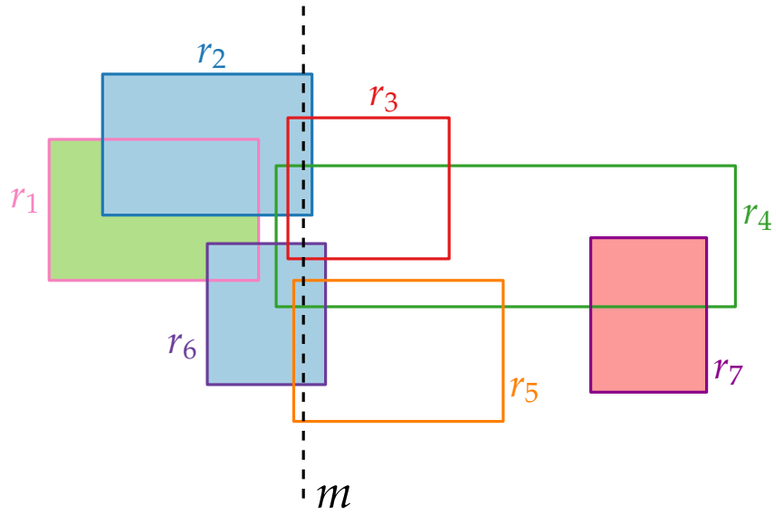
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m

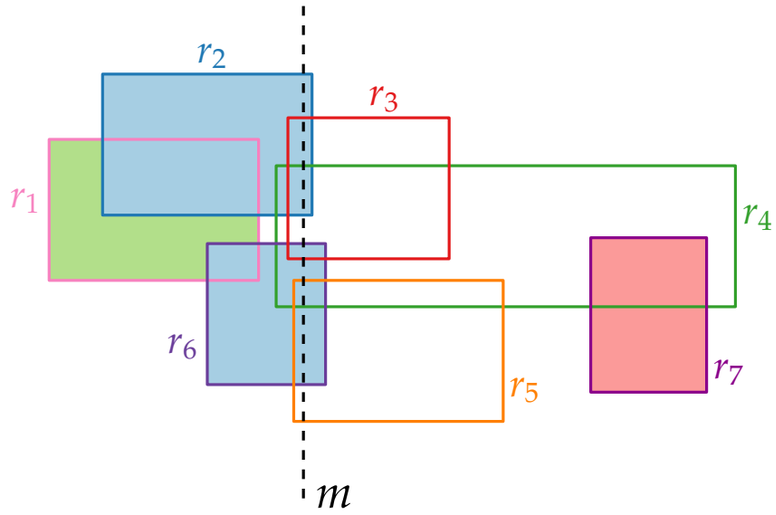
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

  Berechne m, R_ℓ, R_m, R_r (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if $|L_m| \geq |L_\ell| + |L_r|$ **then**
    return L_m
  return $L_\ell \cup L_r$

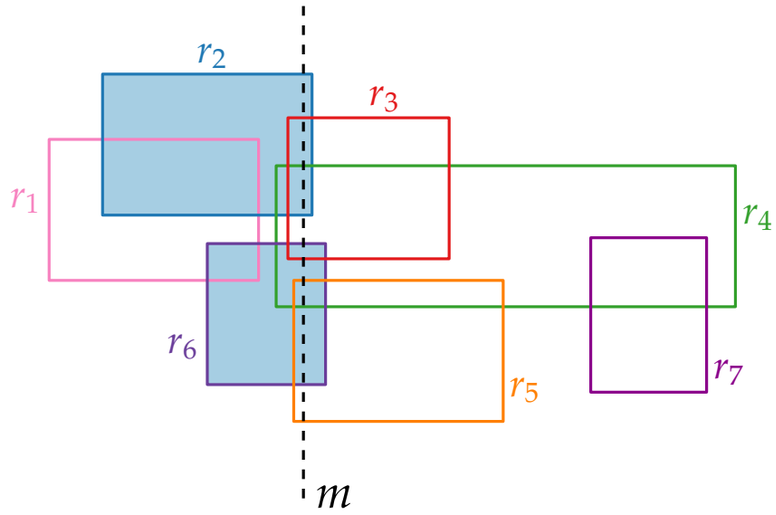
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

  Berechne m, R_ℓ, R_m, R_r (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if $|L_m| \geq |L_\ell| + |L_r|$ **then**
    return L_m
  return $L_\ell \cup L_r$

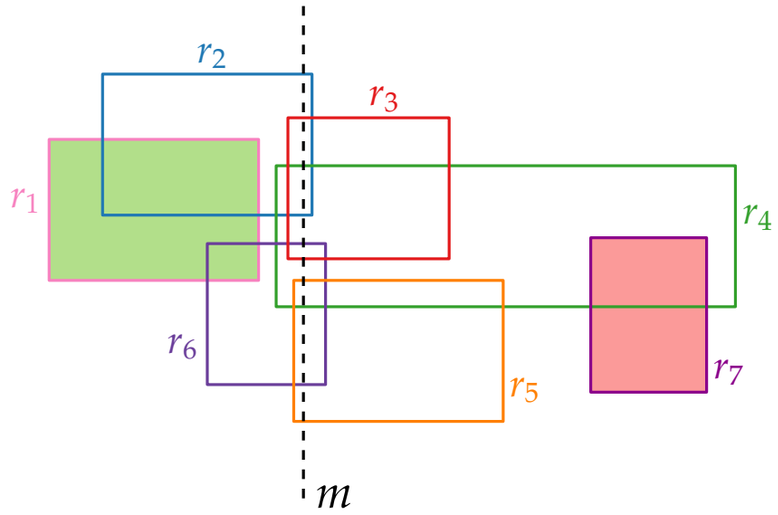
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

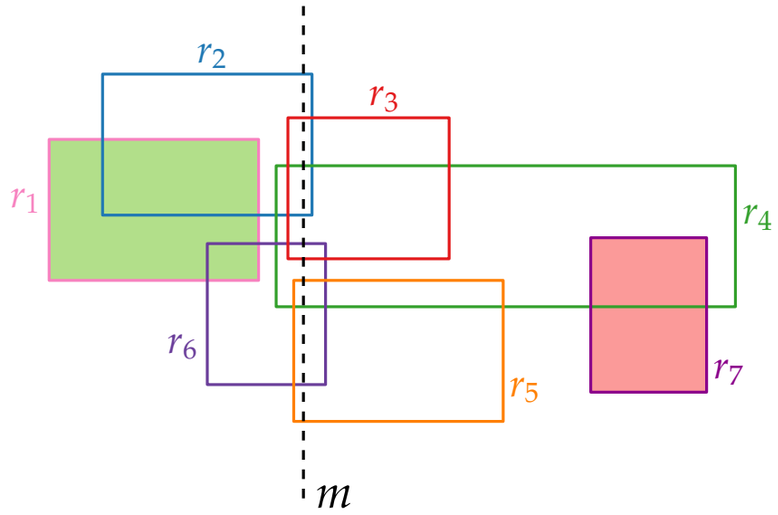
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m




```
MaxNumberRect(R)
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Laufzeit?

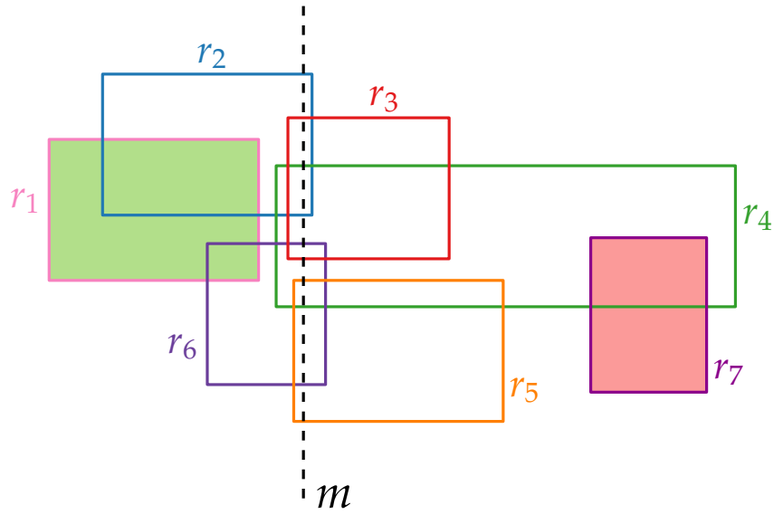
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$

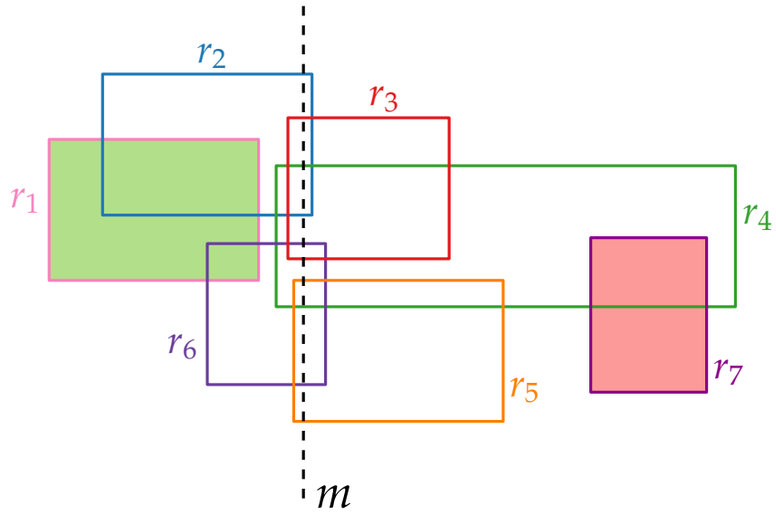
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

  Berechne m, R_ℓ, R_m, R_r (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if $|L_m| \geq |L_\ell| + |L_r|$ **then**
    return L_m
  return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$

$\mathcal{O}(n)$

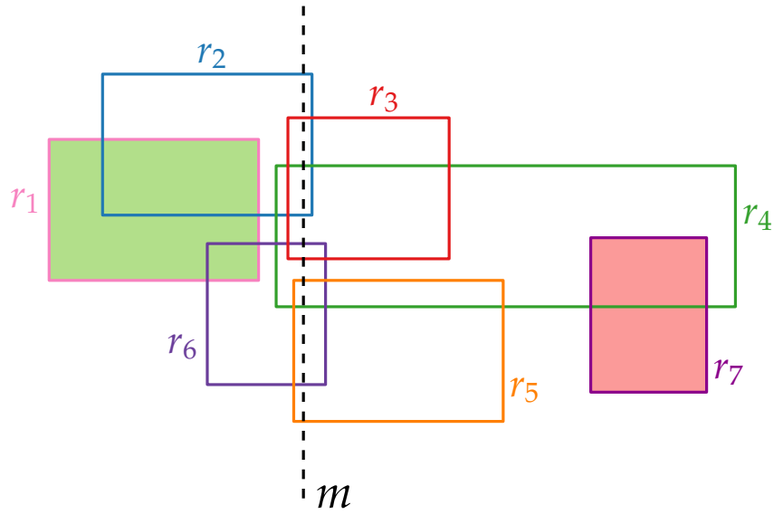
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

  Berechne m, R_ℓ, R_m, R_r (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if $|L_m| \geq |L_\ell| + |L_r|$ **then**
    return L_m
  return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$

$\mathcal{O}(n)$

$\leq T(n/2)$

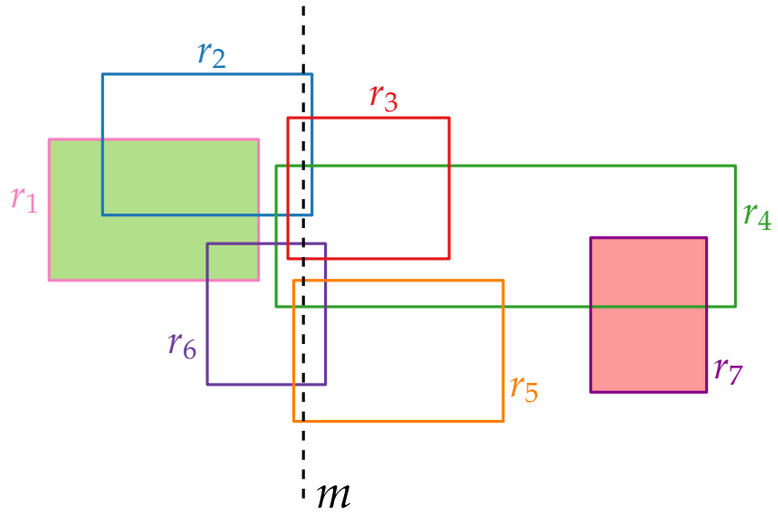
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$
$\mathcal{O}(n)$
$\leq T(n/2)$
$\leq T(n/2)$

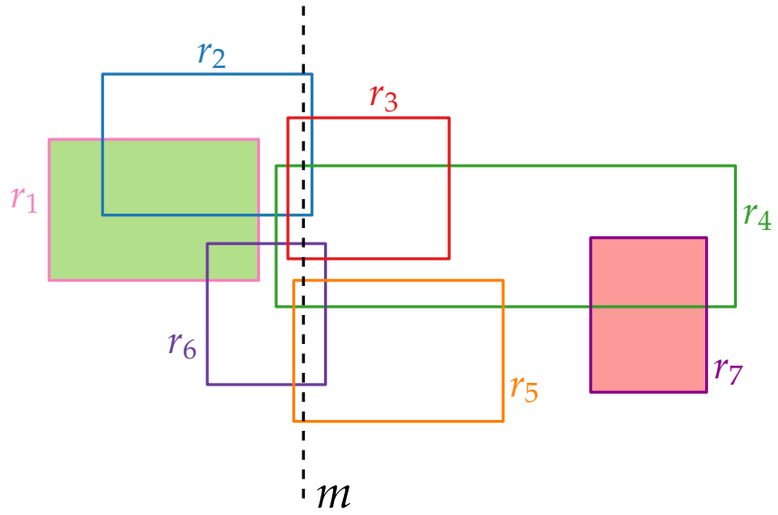
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m



MaxNumberRect(R)

Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Laufzeit?

$\mathcal{O}(n)$
$\mathcal{O}(n)$
$\leq T(n/2)$
$\leq T(n/2)$

$$T(n) \leq 2T(n/2) + \mathcal{O}(n)$$

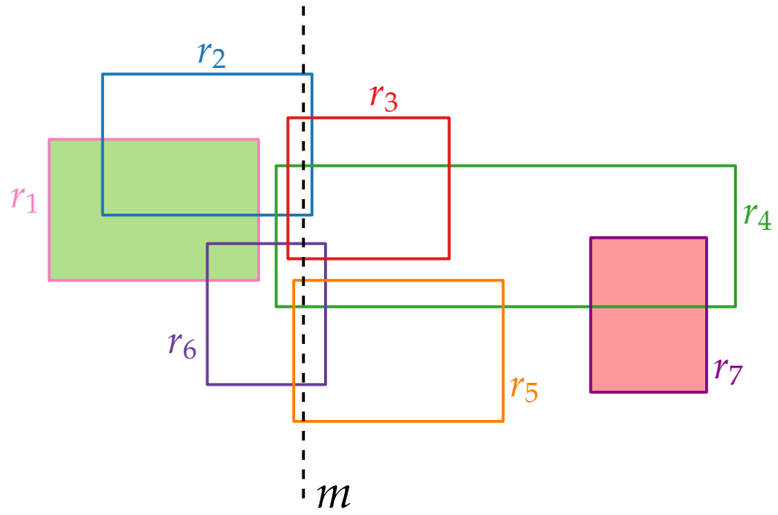
Allgemeinfall

MaxNumber 4P

Geg.: Eine Menge $R = R_1 \cup \ldots \cup R_n$ von Rechtecken

Ges.: Eine **maximale Teilmenge** aus R ohne Überlappung

- Bestimme Median m von $\bigcup_{r \in R} \{r.\text{xMin}, r.\text{xMax}\}$
- $R_m \subseteq R$: Rechtecke auf der Vertikalen durch m
- $R_\ell \subseteq R$: Rechtecke links der Vertikalen durch m
- $R_r \subseteq R$: Rechtecke rechts der Vertikalen durch m




```
MaxNumberRect(R)
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Laufzeit?

$\mathcal{O}(n)$
$\mathcal{O}(n)$
$\leq T(n/2)$
$\leq T(n/2)$

(wie MergeSort)

$$T(n) \leq 2T(n/2) + \mathcal{O}(n) = \mathcal{O}(n \log n)$$

Approximationsfaktor

MaxNumber 4P

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|$

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis.

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r1}
  if n = 2 then
      if r1 ∩ r2 then return {r1}
      else return {r1, r2}
  Berechne m, Rℓ, Rm, Rr
  Lm = GreedyMaxNumber(Rm)
  Lℓ = MaxNumberRect(Rℓ)
  Lr = MaxNumberRect(Rr)
  if |Lm| ≥ |Lℓ| + |Lr| then
      return Lm
  return Lℓ ∪ Lr
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*|$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$)

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell|$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*| / \log(n/2)$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r1}
  if n = 2 then
      if r1 ∩ r2 then return {r1}
      else return {r1, r2}
  Berechne m, Rℓ, Rm, Rr        (optimal)
  Lm = GreedyMaxNumber(Rm)
  Lℓ = MaxNumberRect(Rℓ)
  Lr = MaxNumberRect(Rr)
  if |Lm| ≥ |Lℓ| + |Lr| then
      return Lm
  return Lℓ ∪ Lr
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r1}
  if n = 2 then
      if r1 ∩ r2 then return {r1}
      else return {r1, r2}
  Berechne m, Rℓ, Rm, Rr          (optimal)
  Lm = GreedyMaxNumber(Rm)
  Lℓ = MaxNumberRect(Rℓ)
  Lr = MaxNumberRect(Rr)
  if |Lm| ≥ |Lℓ| + |Lr| then
      return Lm
  return Lℓ ∪ Lr
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

  ➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\}$

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Rände links von m)

➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*| / \log(n/2) \geq |L^* \cap R_\ell| / (\log n - 1)$

➡ $|L_r| \geq |L^* \cap R_r| / (\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max \left\{ \quad , \quad \right\}$

MaxNumberRect(R)

```
if n = 0 then return ∅
if n = 1 then return {r1}
if n = 2 then
    if r1 ∩ r2 then return {r1}
    else return {r1, r2}
Berechne m, Rℓ, Rm, Rr          [optimal]
Lm = GreedyMaxNumber(Rm)
Lℓ = MaxNumberRect(Rℓ)
Lr = MaxNumberRect(Rr)
if |Lm| ≥ |Lℓ| + |Lr| then
    return Lm
return Lℓ ∪ Lr
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

  ➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max\left\{|L^* \cap R_m|, \right\}$

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r1}
    if n = 2 then
        if r1 ∩ r2 then return {r1}
        else return {r1, r2}
    Berechne m, Rℓ, Rm, Rr        (optimal)
    Lm = GreedyMaxNumber(Rm)
    Lℓ = MaxNumberRect(Rℓ)
    Lr = MaxNumberRect(Rr)
    if |Lm| ≥ |Lℓ| + |Lr| then
        return Lm
    return Lℓ ∪ Lr
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

  ➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max\left\{|L^* \cap R_m|, \frac{|L^* \cap R_\ell| + |L^* \cap R_r|}{\log n - 1}\right\}$

MaxNumberRect(R)

```
if n = 0 then return ∅
if n = 1 then return {r_1}
if n = 2 then
    if r_1 ∩ r_2 then return {r_1}
    else return {r_1, r_2}
Berechne m, R_ℓ, R_m, R_r        (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if |L_m| ≥ |L_ℓ| + |L_r| then
    return L_m
return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

  ➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max\left\{|L^* \cap R_m|, \frac{|L^* \cap R_\ell| + |L^* \cap R_r|}{\log n - 1}\right\}$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

(IA) MaxNumberRect ist optimal für $n \leq 2$.

(IV) Satz gilt für alle $|R| < n$.

(IS) Sei $|R| = n$.

Seien L_m^*, L_ℓ^*, L_r^* optimale Lösungen für R_m, R_ℓ, R_r.

- $|L_m| = |L_m^*| \geq |L^* \cap R_m|$
- $|R_\ell| \leq n/2$ (n vertikale Ränder links von m)

  ➡ $|L_\ell| \underset{\text{(IV)}}{\geq} |L_\ell^*|/\log(n/2) \geq |L^* \cap R_\ell|/(\log n - 1)$

  ➡ $|L_r| \geq |L^* \cap R_r|/(\log n - 1)$

- $|L| = \max\{|L_m|, |L_\ell| + |L_r|\} \geq \max\left\{|L^* \cap R_m|, \frac{|L^* \cap R_\ell| + |L^* \cap R_r|}{\log n - 1}\right\}$

  $= |L^*| - |L^* \cap R_m|$

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*| - |L^* \cap R_m|}{\log n - 1}\right\}$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*| - |L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1:

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r        (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{ |L^* \cap R_m|, \frac{|L^*| - |L^* \cap R_m|}{\log n - 1} \right\}$

Fall 1:

```
MaxNumberRect(R)
    if n = 0 then return ∅
    if n = 1 then return {r_1}
    if n = 2 then
        if r_1 ∩ r_2 then return {r_1}
        else return {r_1, r_2}
    Berechne m, R_ℓ, R_m, R_r          (optimal)
    L_m = GreedyMaxNumber(R_m)
    L_ℓ = MaxNumberRect(R_ℓ)
    L_r = MaxNumberRect(R_r)
    if |L_m| ≥ |L_ℓ| + |L_r| then
        return L_m
    return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2:

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

➡ $|L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}$

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*| / \log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*| - |L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*| / \log n$

➡ $|L| \geq |L^* \cap R_m| \geq |L^*| / \log n$

Fall 2: $|L^* \cap R_m| < |L^*| / \log n$

➡ $|L| \geq \frac{|L^*| - |L^* \cap R_m|}{\log n - 1} > \frac{|L^*| - |L^*| / \log n}{\log n - 1}$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n - 1}$$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1}$$

$$= \frac{|L^*|(1-1/\log n)}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n(1-1/\log n)}$$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1}$$
$$= \frac{|L^*|(1-1/\log n)}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n(1-1/\log n)}$$
$$= |L^*|/\log n$$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1}$$
$$= \frac{|L^*|(1-1/\log n)}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n(1-1/\log n)}$$
$$= |L^*|/\log n$$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1}$$

$$= \frac{|L^*|(1-1/\log n)}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n(1-1/\log n)}$$

$$= |L^*|/\log n$$

MaxNumberRect(R)

if $n = 0$ **then return** $\emptyset$
if $n = 1$ **then return** $\{r_1\}$
if $n = 2$ **then**
  if $r_1 \cap r_2$ **then return** $\{r_1\}$
  else return $\{r_1, r_2\}$
Berechne m, R_ℓ, R_m, R_r (optimal)
L_m = GreedyMaxNumber(R_m)
L_ℓ = MaxNumberRect(R_ℓ)
L_r = MaxNumberRect(R_r)
if $|L_m| \geq |L_\ell| + |L_r|$ **then**
  return L_m
return $L_\ell \cup L_r$

Approximationsfaktor

MaxNumber 4P

Sei L^* eine optimale Lösung für R.

Satz. MaxNumberRect(R) findet eine gültige Lösung L mit $|L| \geq |L^*|/\log n$.

Beweis. Durch Induktion über n.

- $|L| \geq \max\left\{|L^* \cap R_m|, \frac{|L^*|-|L^* \cap R_m|}{\log n - 1}\right\}$

Fall 1: $|L^* \cap R_m| \geq |L^*|/\log n$

$\Rightarrow |L| \geq |L^* \cap R_m| \geq |L^*|/\log n$

Fall 2: $|L^* \cap R_m| < |L^*|/\log n$

$$\Rightarrow |L| \geq \frac{|L^*|-|L^* \cap R_m|}{\log n - 1} > \frac{|L^*|-|L^*|/\log n}{\log n - 1}$$

$$= \frac{|L^*|(1-1/\log n)}{\log n - 1} = \frac{|L^*|(1-1/\log n)}{\log n(1-1/\log n)}$$

$$= |L^*|/\log n$$

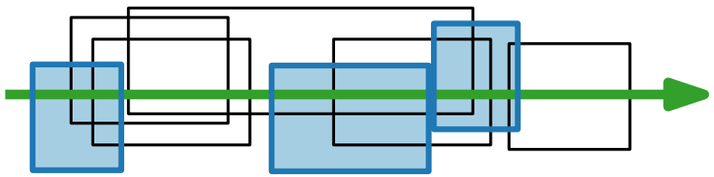
□

```
MaxNumberRect(R)
  if n = 0 then return ∅
  if n = 1 then return {r_1}
  if n = 2 then
      if r_1 ∩ r_2 then return {r_1}
      else return {r_1, r_2}
  Berechne m, R_ℓ, R_m, R_r        (optimal)
  L_m = GreedyMaxNumber(R_m)
  L_ℓ = MaxNumberRect(R_ℓ)
  L_r = MaxNumberRect(R_r)
  if |L_m| ≥ |L_ℓ| + |L_r| then
      return L_m
  return L_ℓ ∪ L_r
```

Universität Trier

Algorithmen für geographische Informationssysteme

6. Vorlesung
Kartenbeschriftung

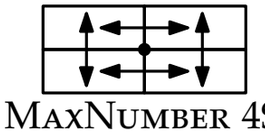


Teil III:
Slider-Modell

Philipp Kindermann

Wintersemester 2024/25

Noch ein Greedy Algorithmus



Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.

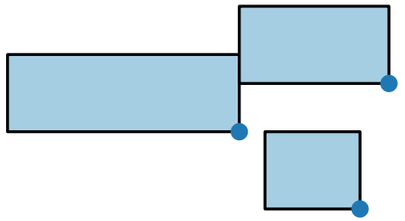
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



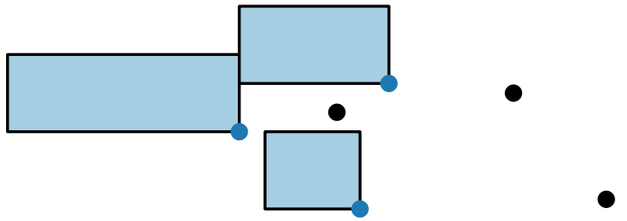
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



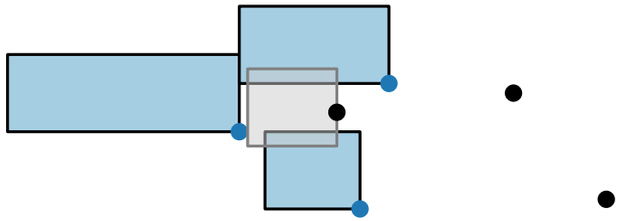
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



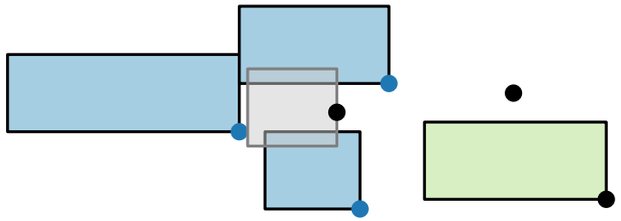
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



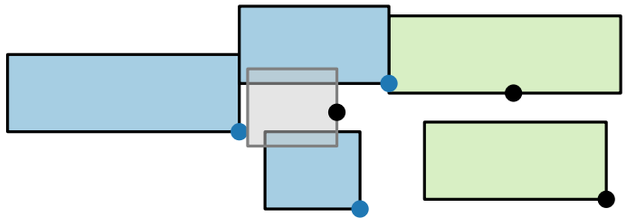
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



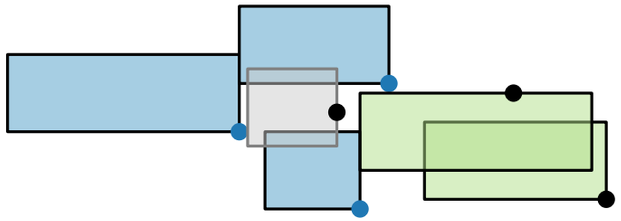
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



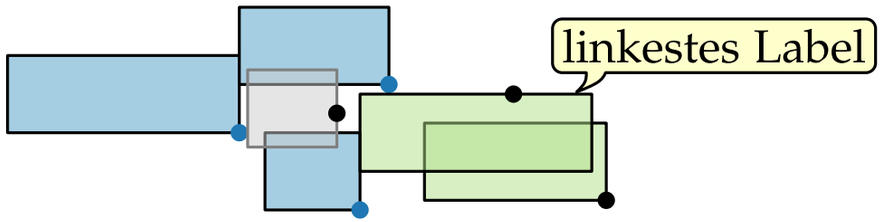
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



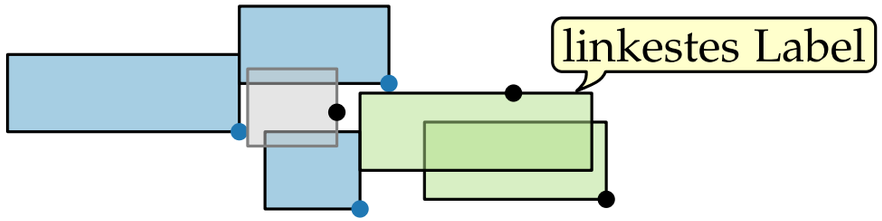
Noch ein Greedy Algorithmus

MaxNumber 4S

Geg.: n Punkte in der Ebene und für jeden Punkt ein Label, repräsentiert als Rechteck (*bounding box*)

Ges.: zulässige Beschriftung für eine **maximale Teilmenge** der Punkte, so dass sich keine zwei Label schneiden (MaxNumber)

Für eine Menge P' von Punkten mit bereits platzierten Labeln heißt Label ℓ für Punkt $p \in P \setminus P'$ **linkestes** Label (in $P \setminus P'$), falls seine rechte Kante unter allen noch überlappungsfrei platzierbaren Labeln am weitesten links liegt.



Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

MaxNumber 4S

GREEDY4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

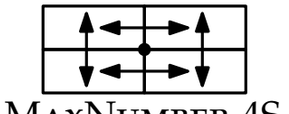
MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.

GREEDY4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor



- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Approximationsfaktor

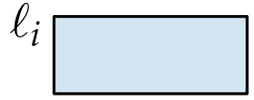
MaxNumber 4S

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?



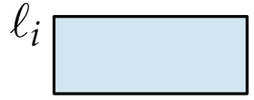
Approximationsfaktor

MaxNumber 4S

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?
- $\ell' \in L_i$



Approximationsfaktor

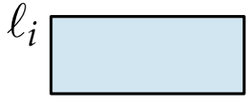
MaxNumber 4S

AGiS

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?
- $\ell' \in L_i$
  ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich



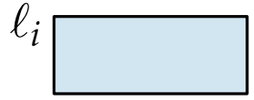
Approximationsfaktor

MaxNumber 4S

GREEDY4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von GREEDY4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Approximationsfaktor

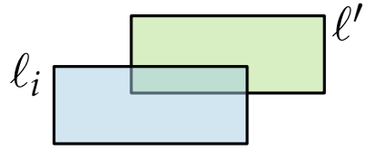
MaxNumber 4S

AGiS

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.
- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Approximationsfaktor

MaxNumber 4S

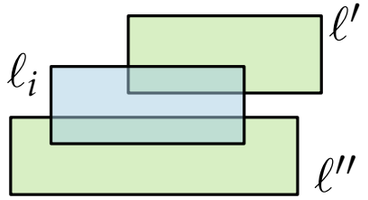
AGiS

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$?
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Approximationsfaktor

MaxNumber 4S

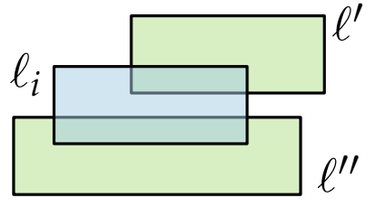
AGiS

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Approximationsfaktor

MaxNumber 4S

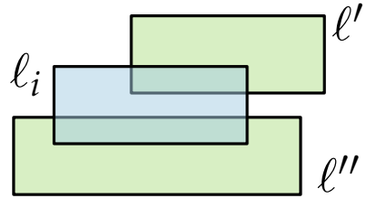
AGiS

- Angenommen, alle Labels haben Höhe 1.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

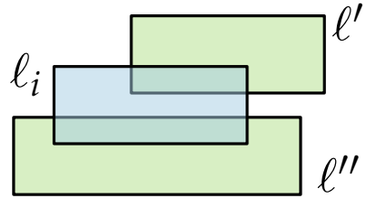
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

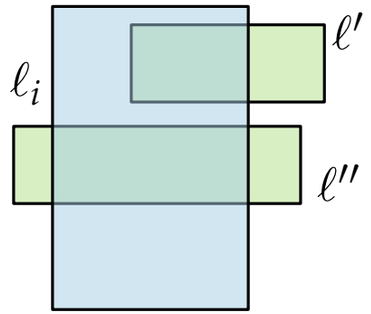
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

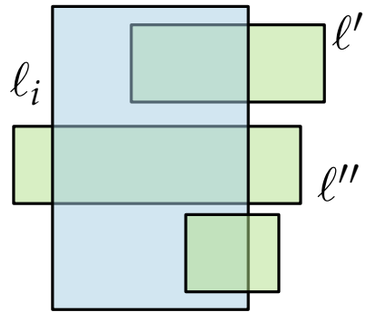
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

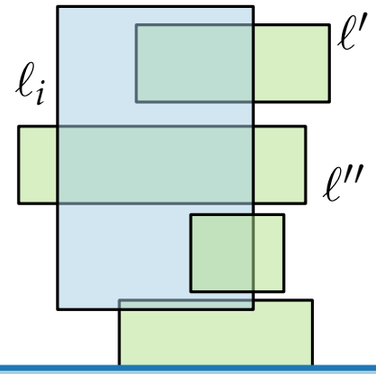
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

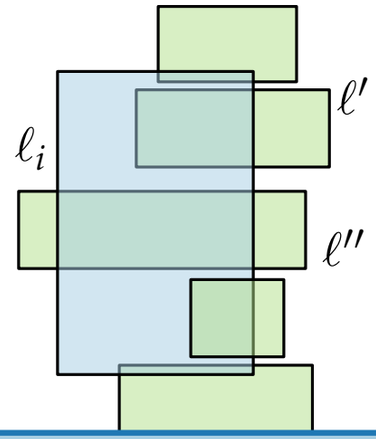
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq 2$!
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

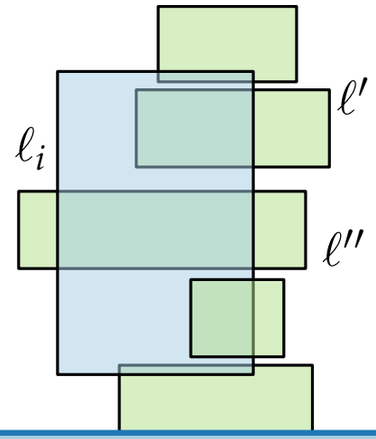
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq$ ~~2~~ $m + 1$
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Approximationsfaktor

MaxNumber 4S

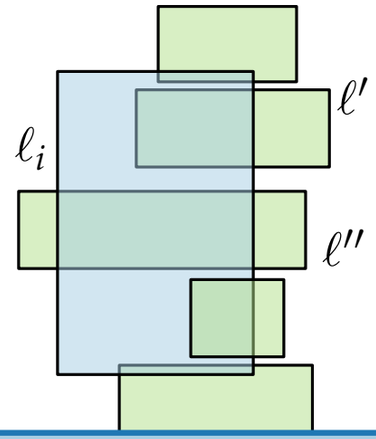
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq$ ~~2~~! $m + 1$
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Satz. Für Labels der Höhen in $[1, m]$ berechnet Greedy4S eine Faktor-$1/(m+1)$-Approximation.

Approximationsfaktor

MaxNumber 4S

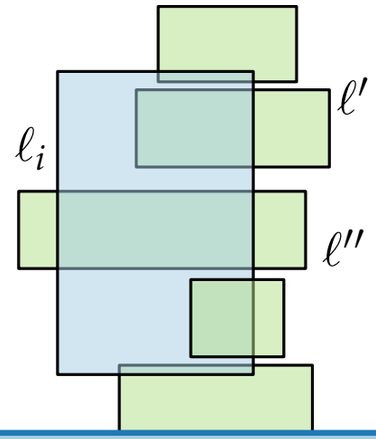
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq$ ~~2~~! $m + 1$
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Laufzeit?

Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Satz. Für Labels der Höhen in $[1, m]$ berechnet Greedy4S eine Faktor-$1/(m+1)$-Approximation.

Approximationsfaktor

MaxNumber 4S

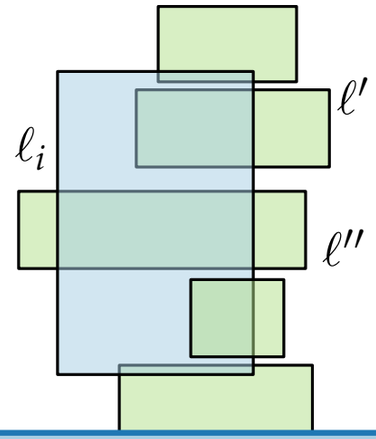
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq$ ~~2~~! $m + 1$
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Laufzeit? $\mathcal{O}(n^3)$.

Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Satz. Für Labels der Höhen in $[1, m]$ berechnet Greedy4S eine Faktor-$1/(m+1)$-Approximation.

Approximationsfaktor

MaxNumber 4S

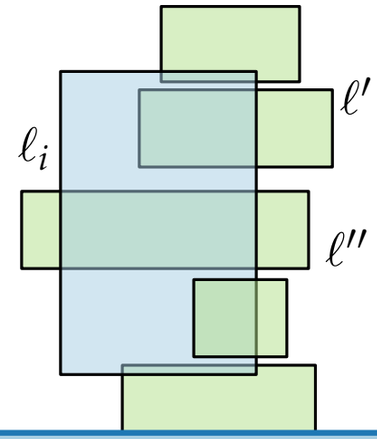
$\in [1, m]$

- Angenommen, alle Labels haben Höhe ~~1~~.
- Sei L Greedy Lösung, L^* optimale Lösung.
- Betrachte Labels $\ell_1, \ldots, \ell_{|L|}$ in der Reihenfolge wie von Greedy4S ausgewählt.

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Jedes Label ℓ_i schneidet mindestens ein Label in L^* (sonst L^* nicht optimal).
- Sei L_i die Menge der Label in L^*, die ℓ_i schneiden, aber nicht $\ell_1, \ldots, \ell_{i-1}$.
- Wie groß ist $|L_i|$? ➡ $|L_i| \leq$ ~~2~~! $m + 1$
- $\ell' \in L_i$
  - ➡ $\ell_1, \ldots, \ell_{i-1}, \ell'$ überlappungsfrei
  - ➡ ℓ' **rechter** als ℓ_i (wegen Greedy Wahl)



Laufzeit? $\mathcal{O}(n^3)$. Schneller möglich?

Satz. Für Labels der Höhe 1 berechnet Greedy4S eine Faktor-1/2-Approximation.

Satz. Für Labels der Höhen in $[1, m]$ berechnet Greedy4S eine Faktor-$1/(m+1)$-Approximation.

Notation

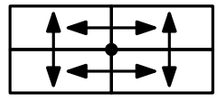
MaxNumber 4S

GREEDY4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.



MaxNumber 4S

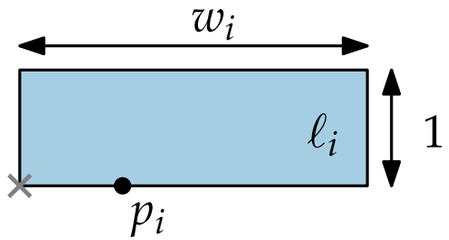
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



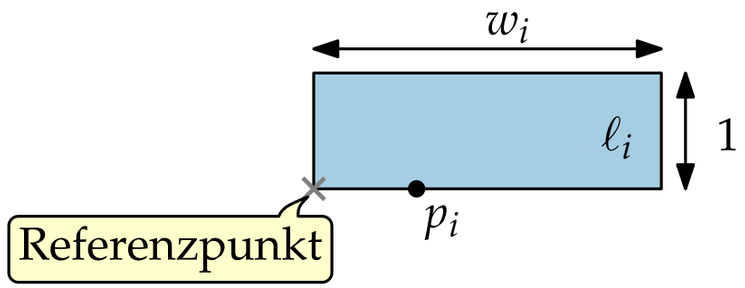
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



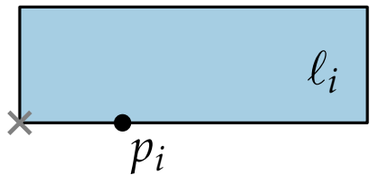
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



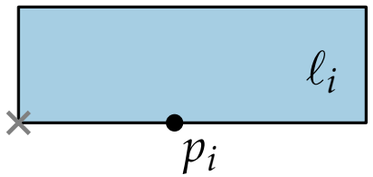
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



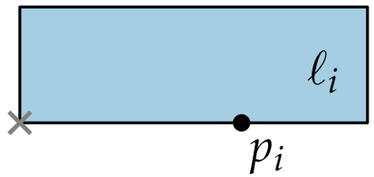
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S

Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S

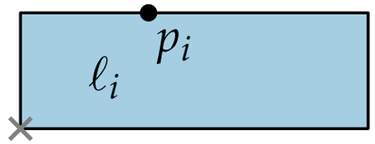
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



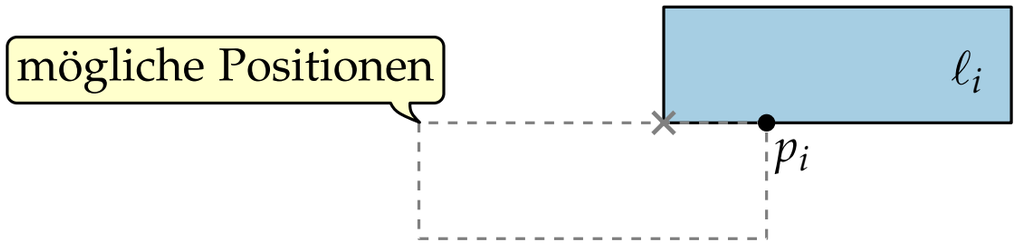
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



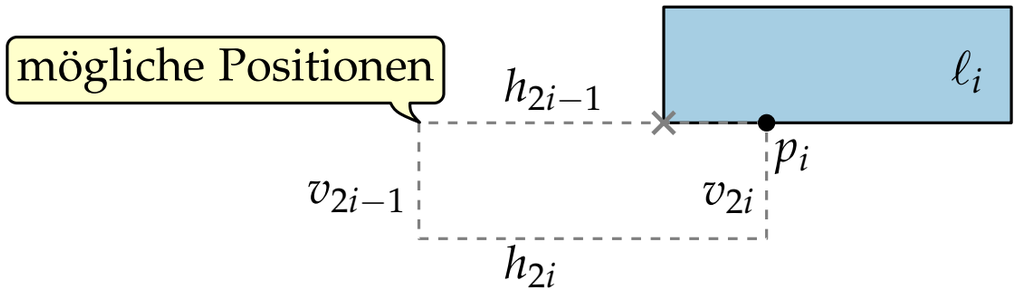
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



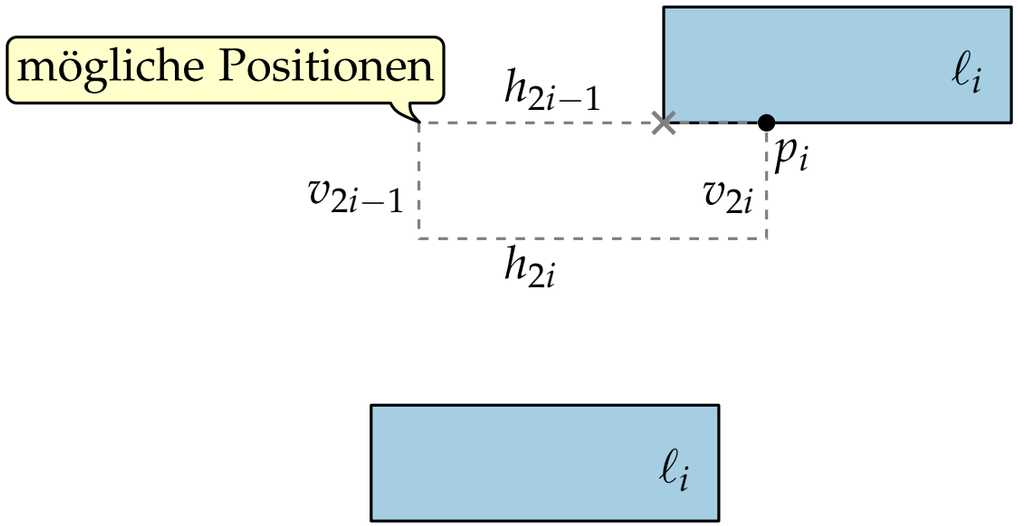
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



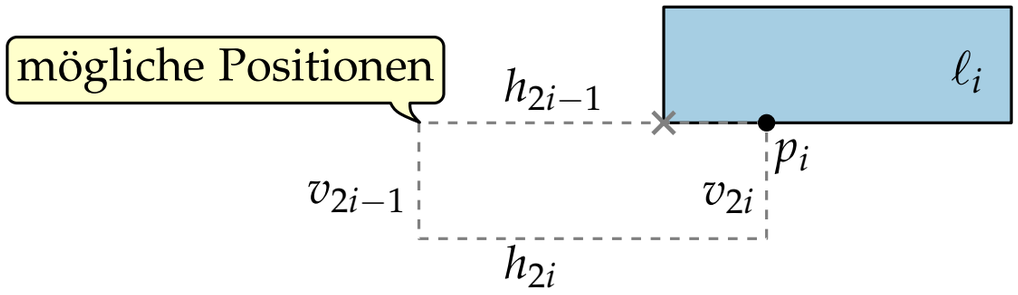
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



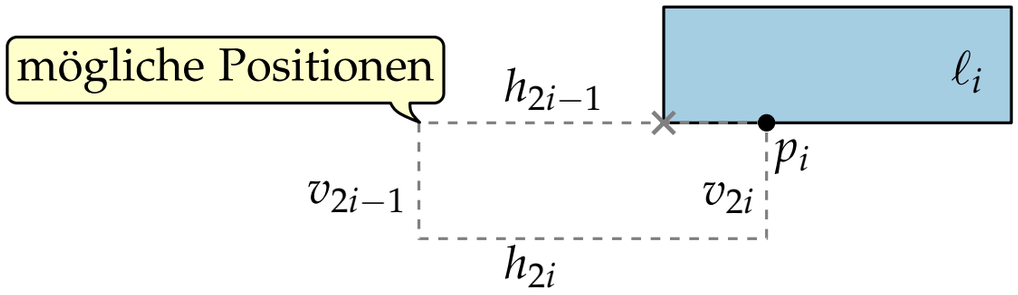
Greedy4S(P, L)

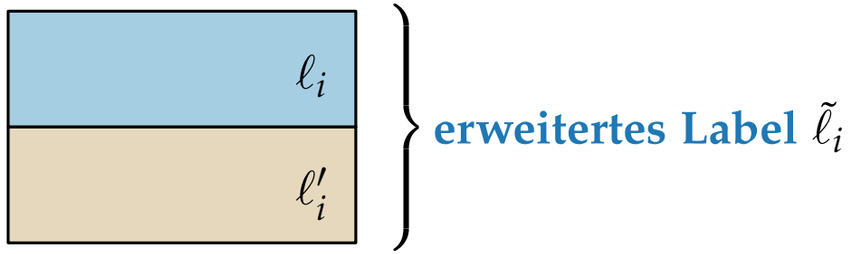
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





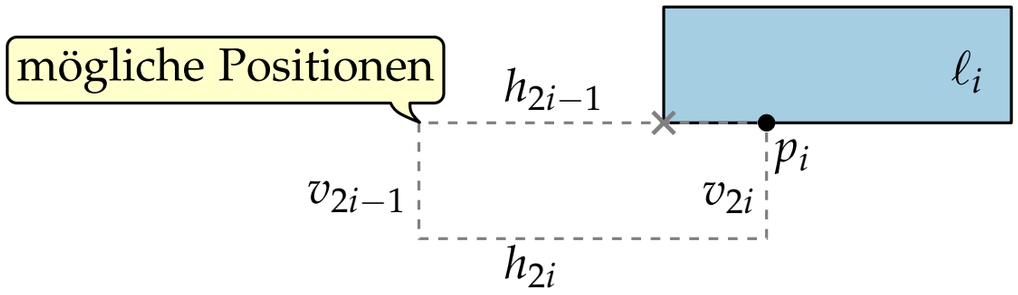
GREEDY4S(P, L)

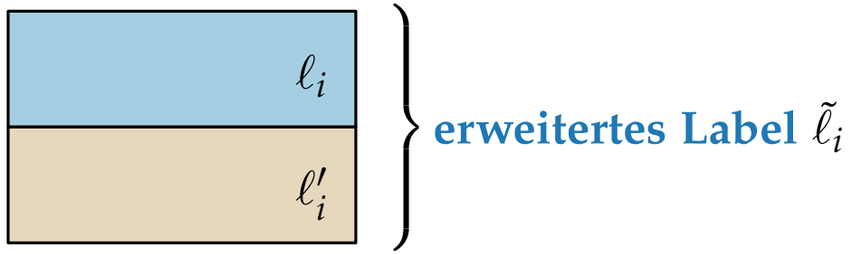
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

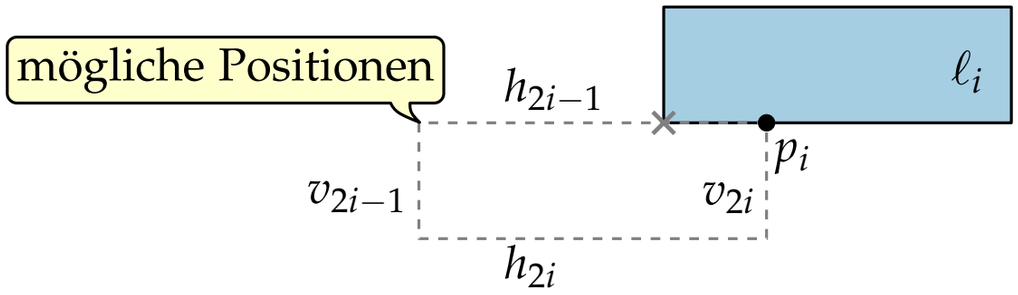
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

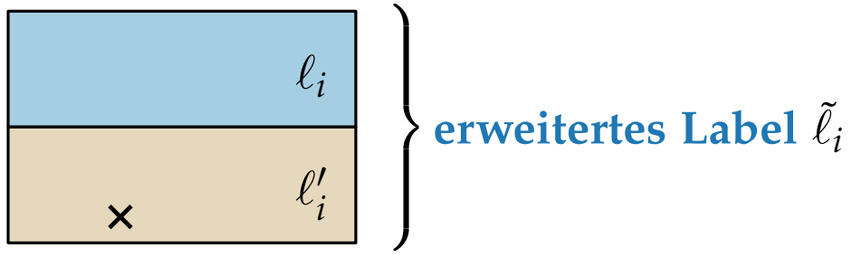
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

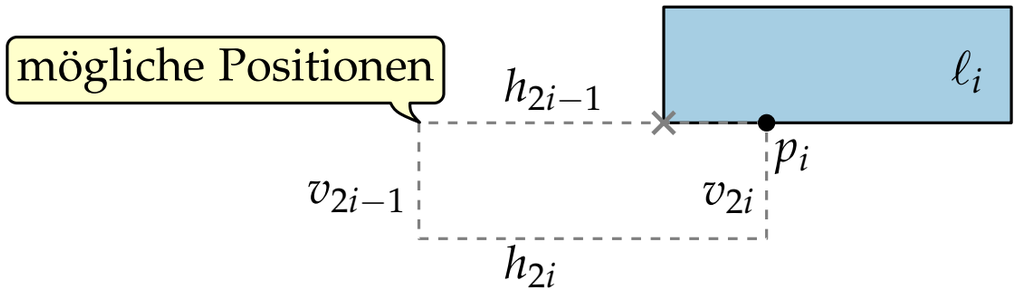
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

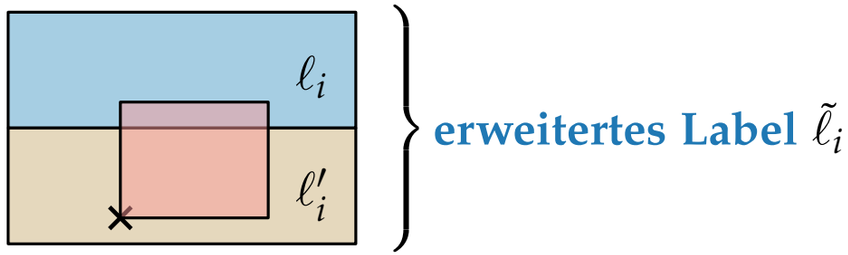
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

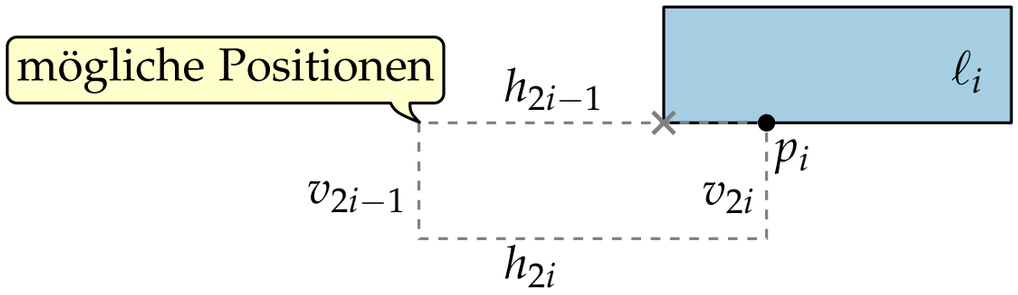
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

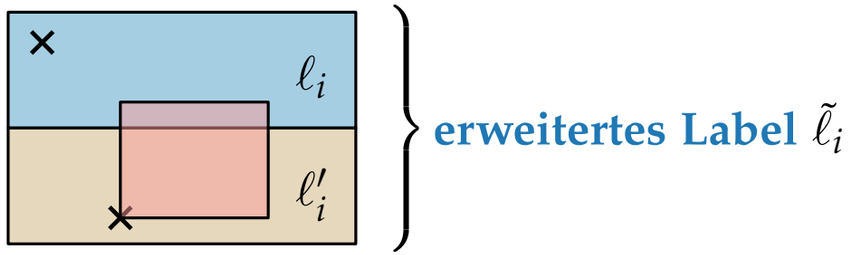
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

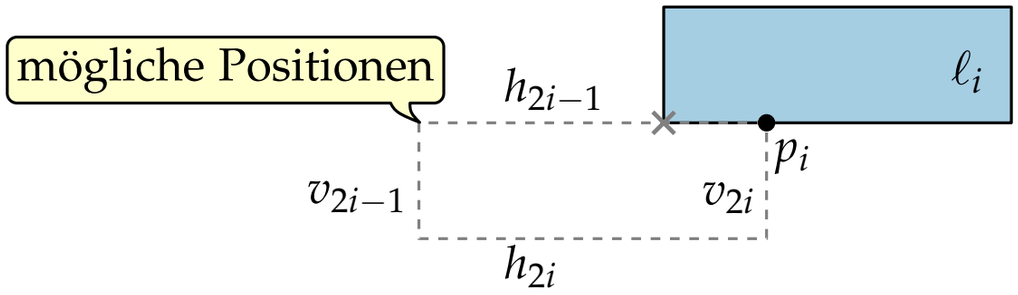
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

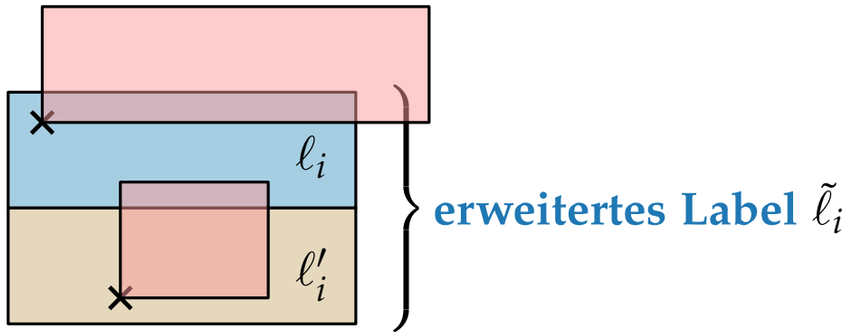
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

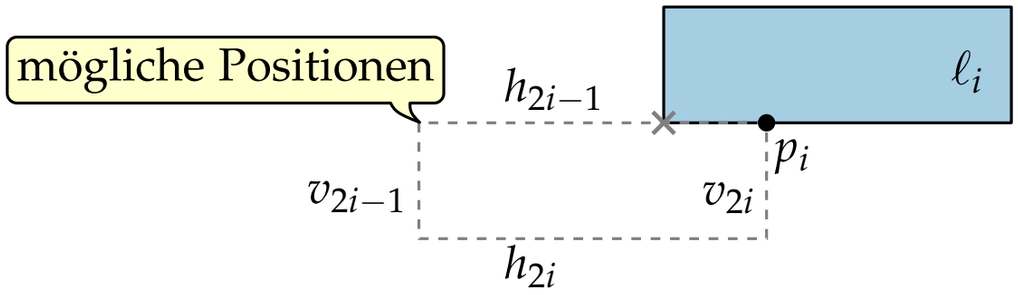
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

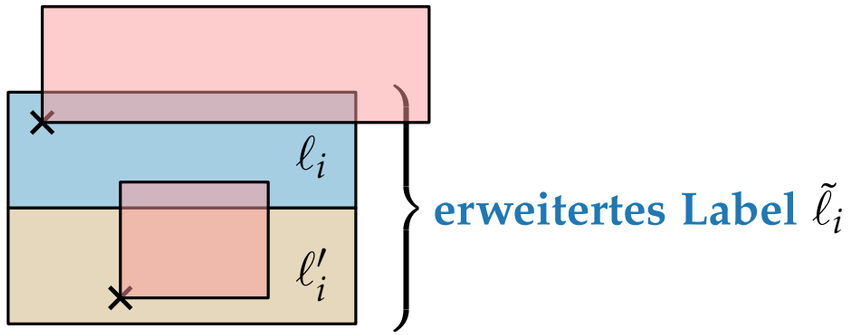
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





GREEDY4S(P, L)

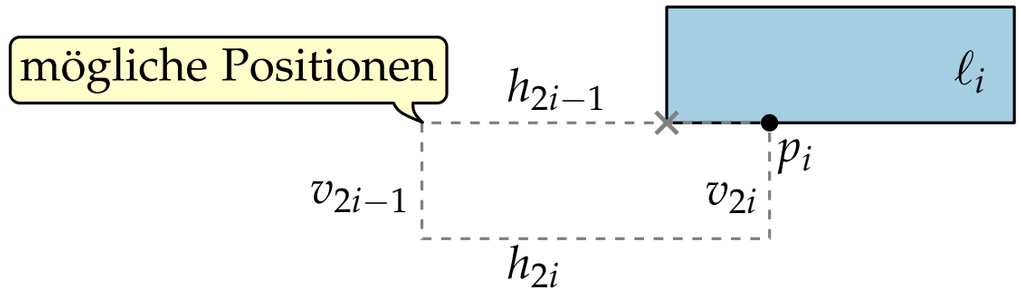
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

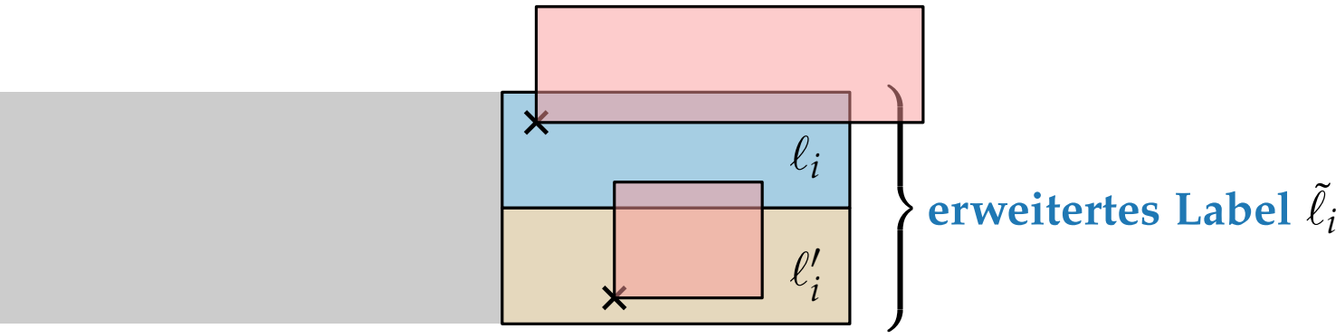
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S





Greedy4S(P, L)

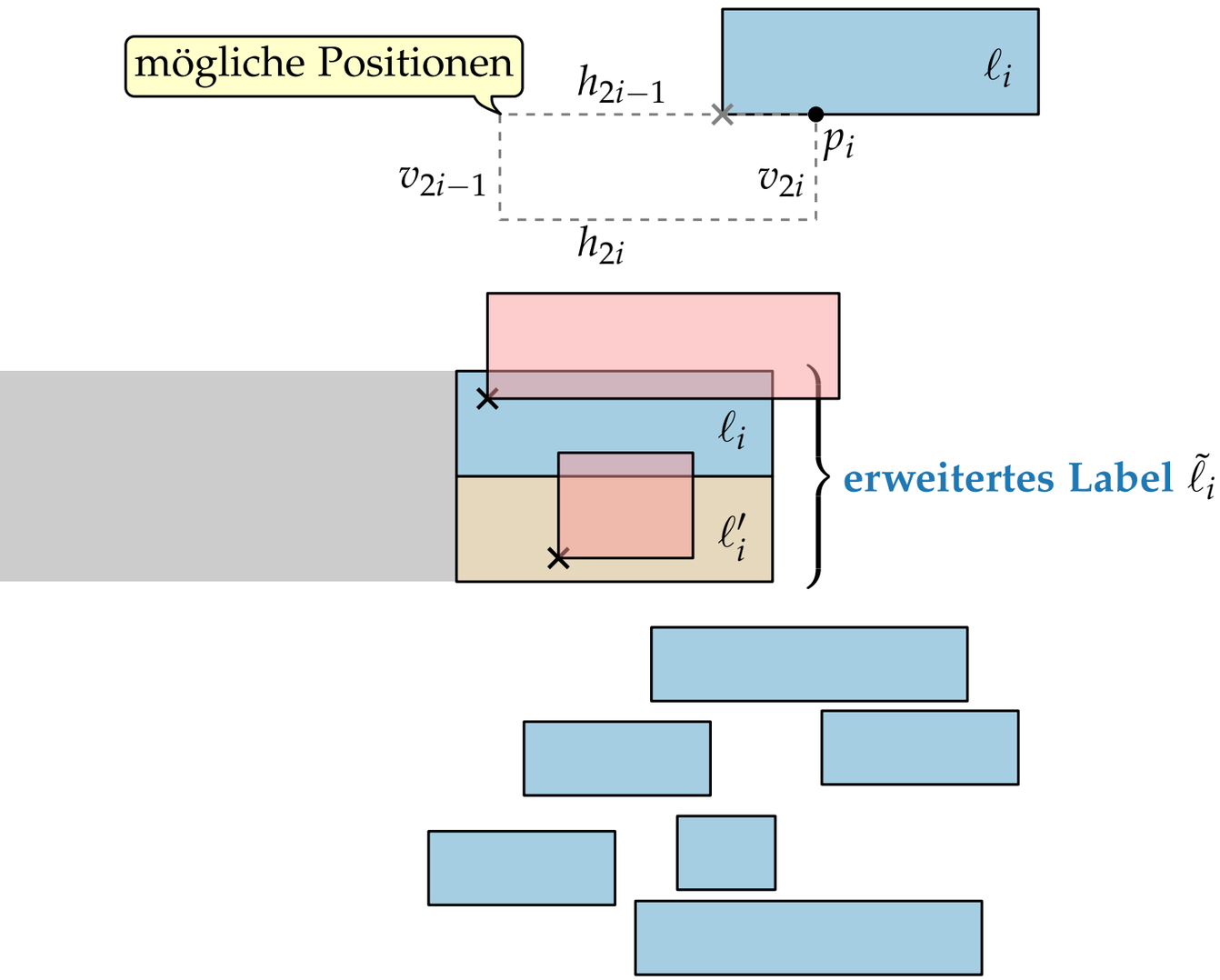
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S



Greedy4S(P, L)

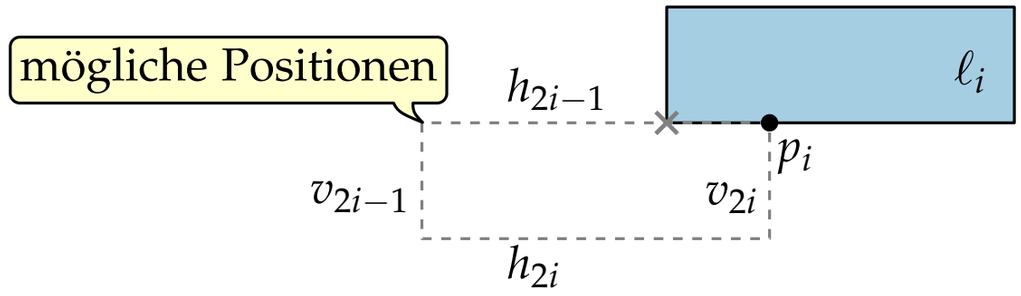
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

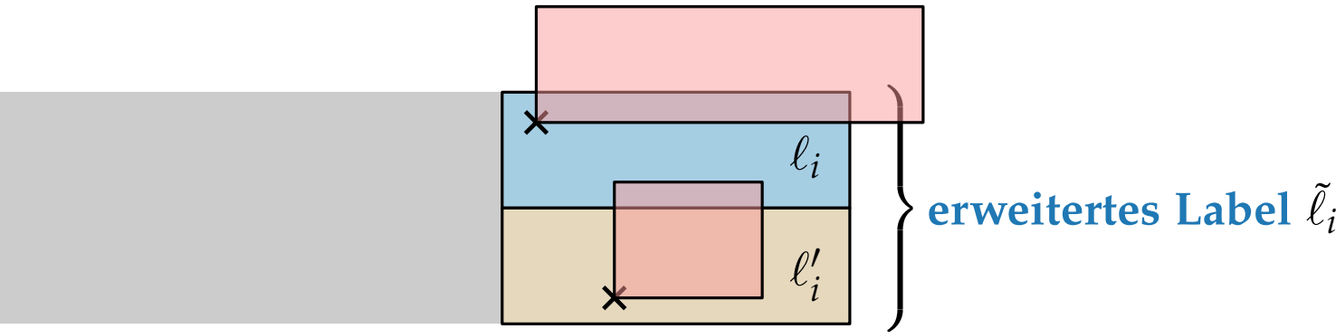
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

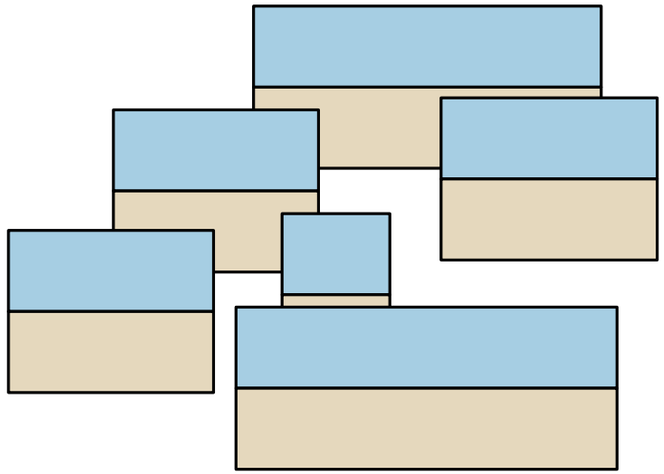
Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S







GREEDY4S(P, L)

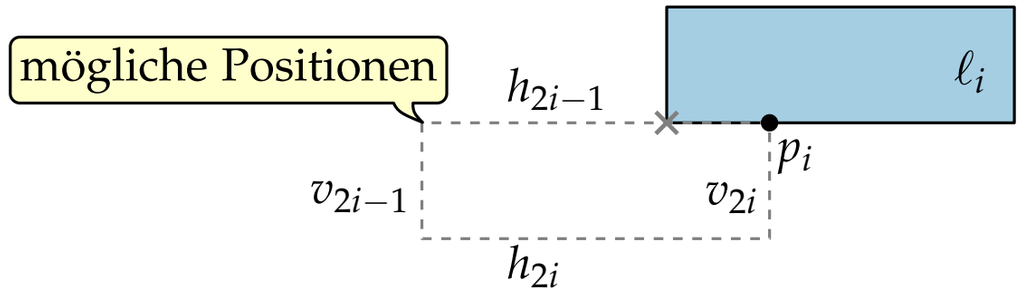
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

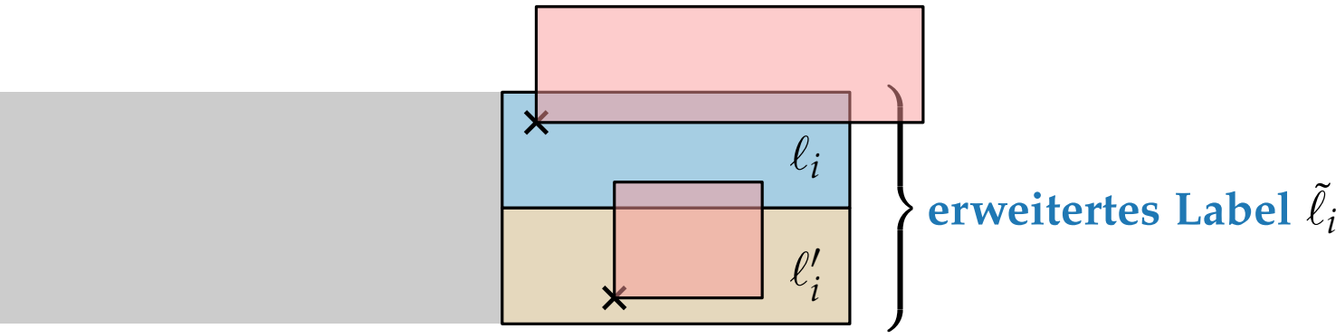
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

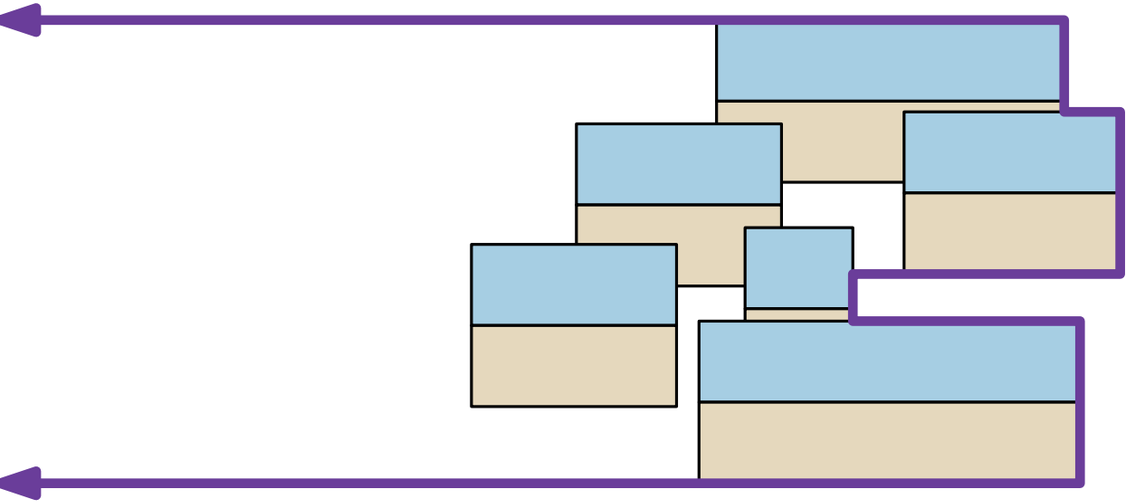
Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S







GREEDY4S(P, L)

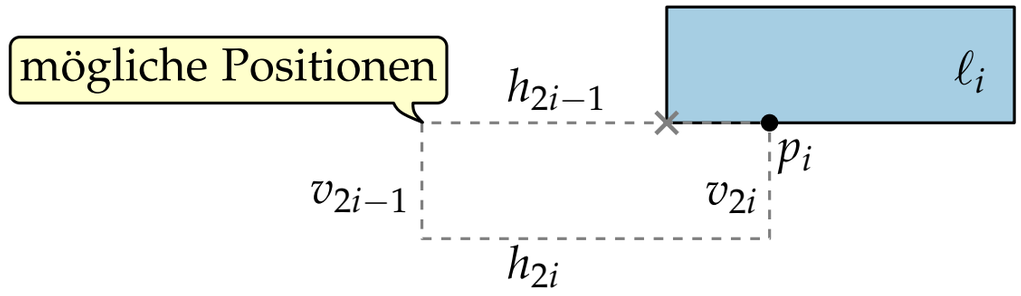
while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

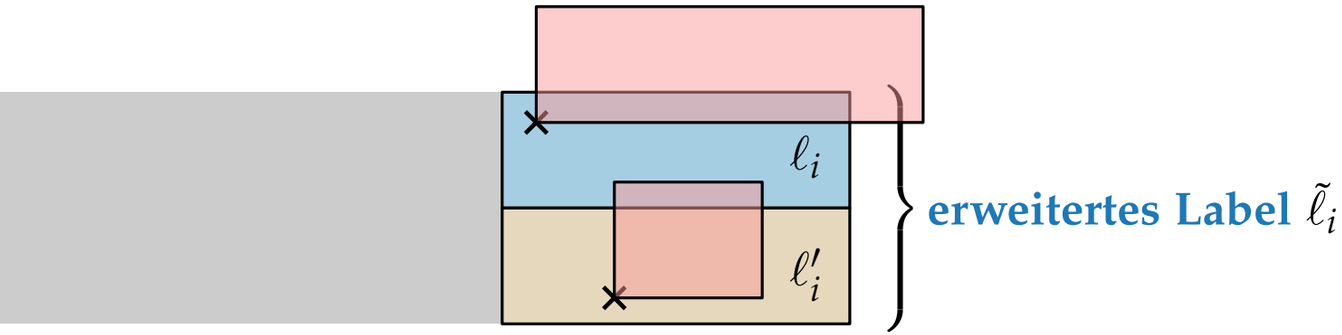
- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

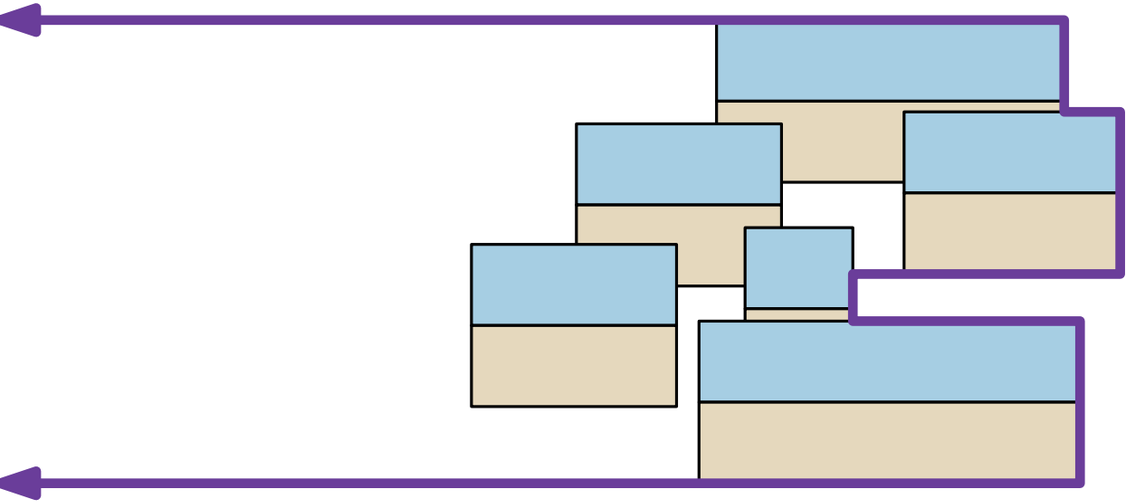
Notation

Wir betrachten nur Fall mit Labelhöhe 1.

MaxNumber 4S







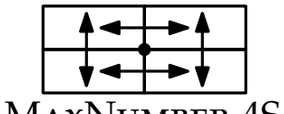
Greedy4S(P, L)

while linkestes Label ℓ existiert **do**
  platziere ℓ linkest möglich

- Kein Referenzpunkt darf in $\tilde{\ell}_i$ liegen.
- Da immer linkestes Label platziert wird, darf kein Referenzpunkt links von $\tilde{\ell}_i$ liegen.

Grenze F als *right envelope* der platzierten erweiterten Label.

Grenze und linkestes Label

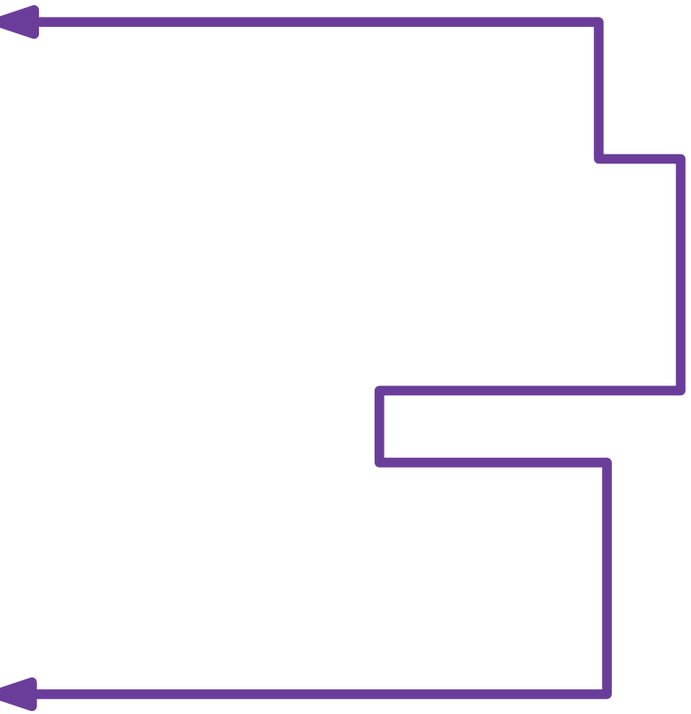


Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

Grenze und linkestes Label

MaxNumber 4S

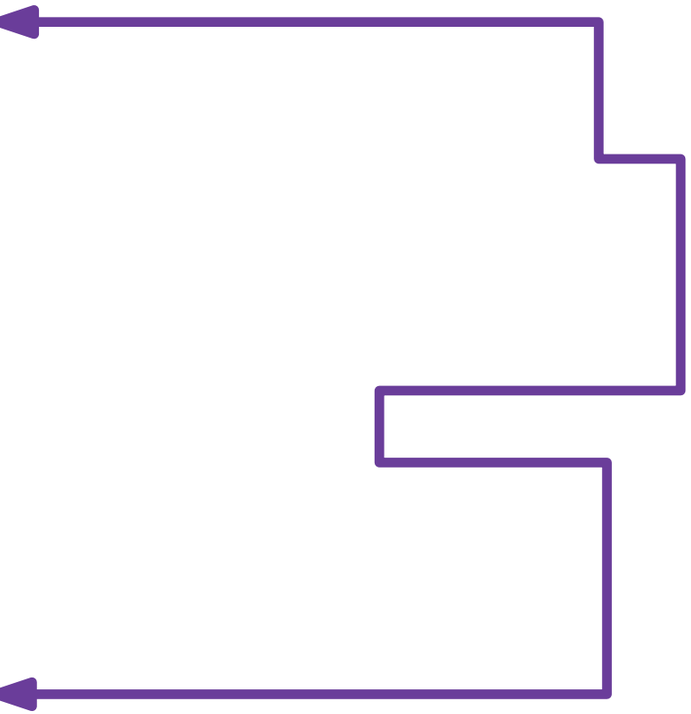
Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

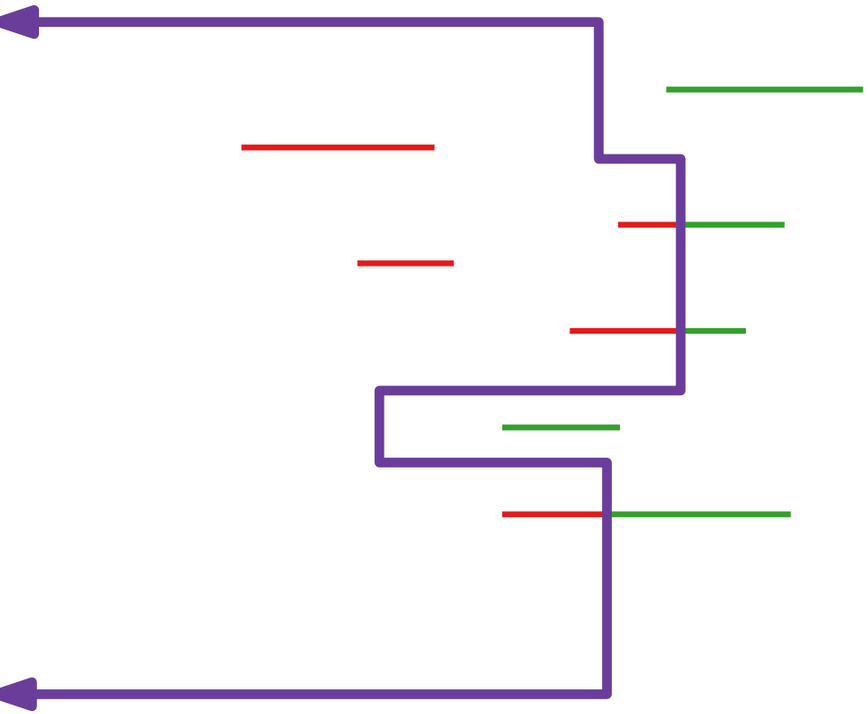


- Menge H der horizontalen Strecken

Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

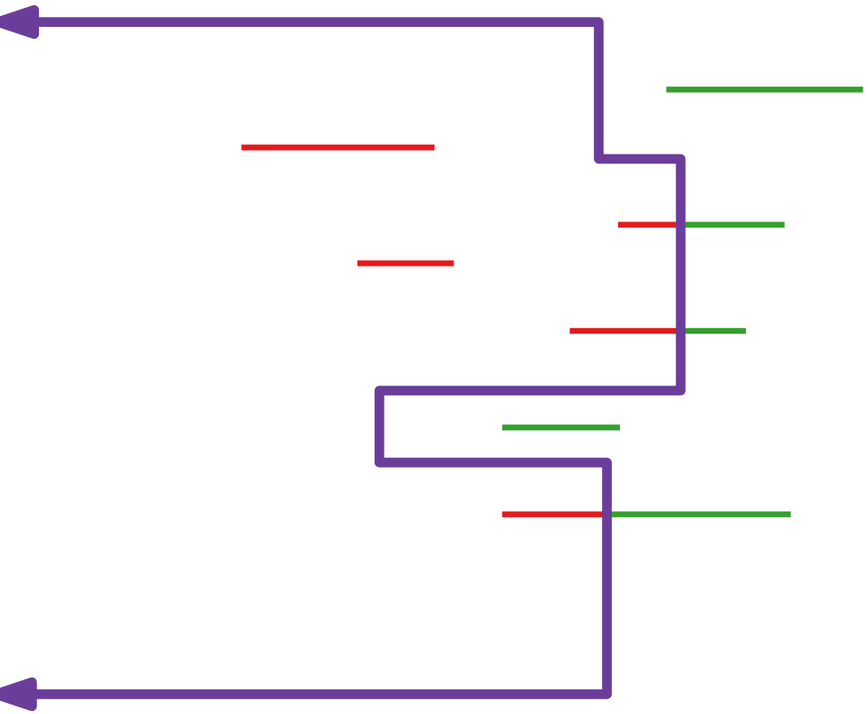


- Menge H der horizontalen Strecken

Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

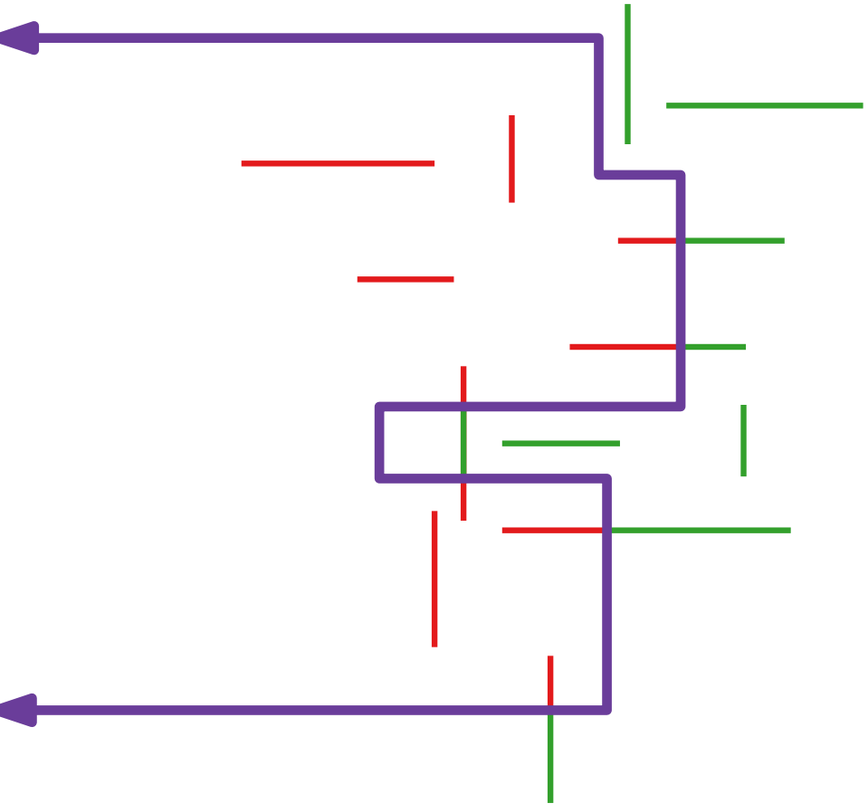


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken

Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

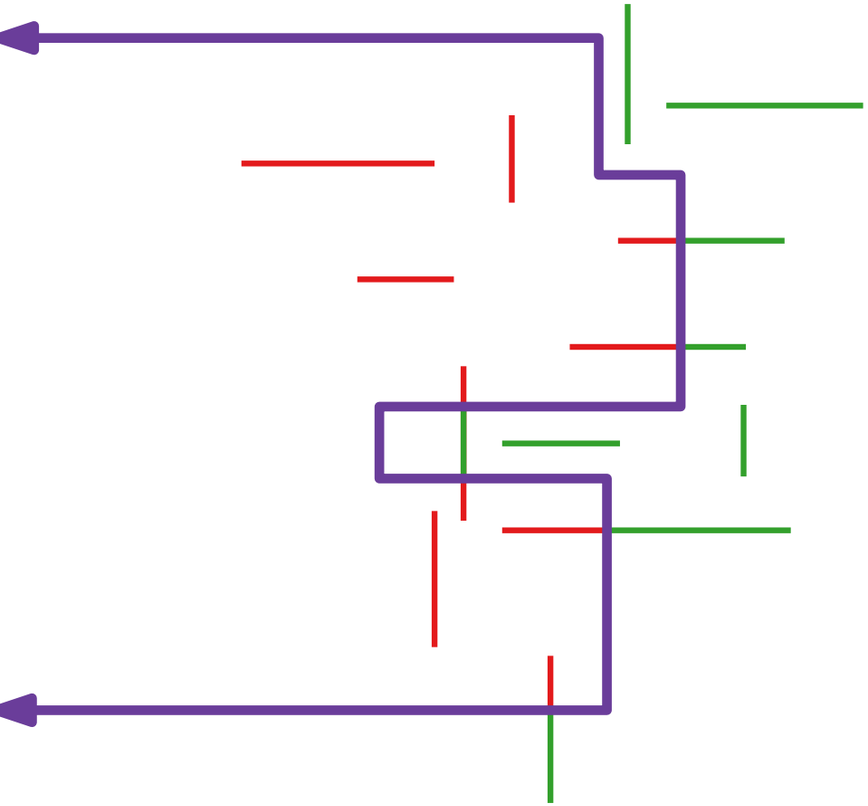


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken

Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

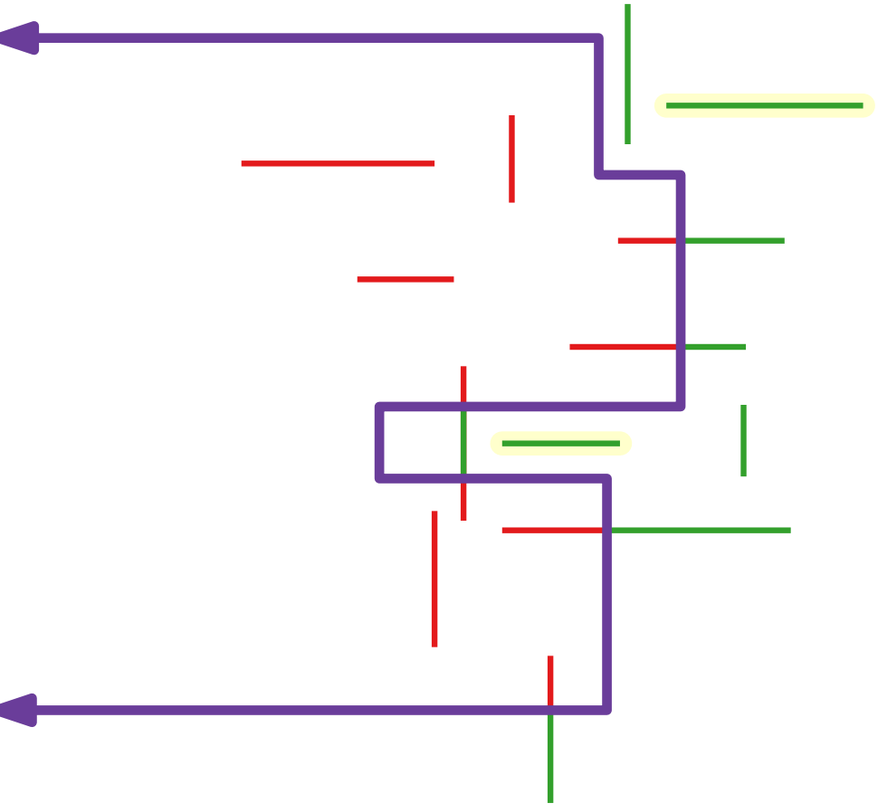


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F

Grenze und linkestes Label

MaxNumber 4S

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

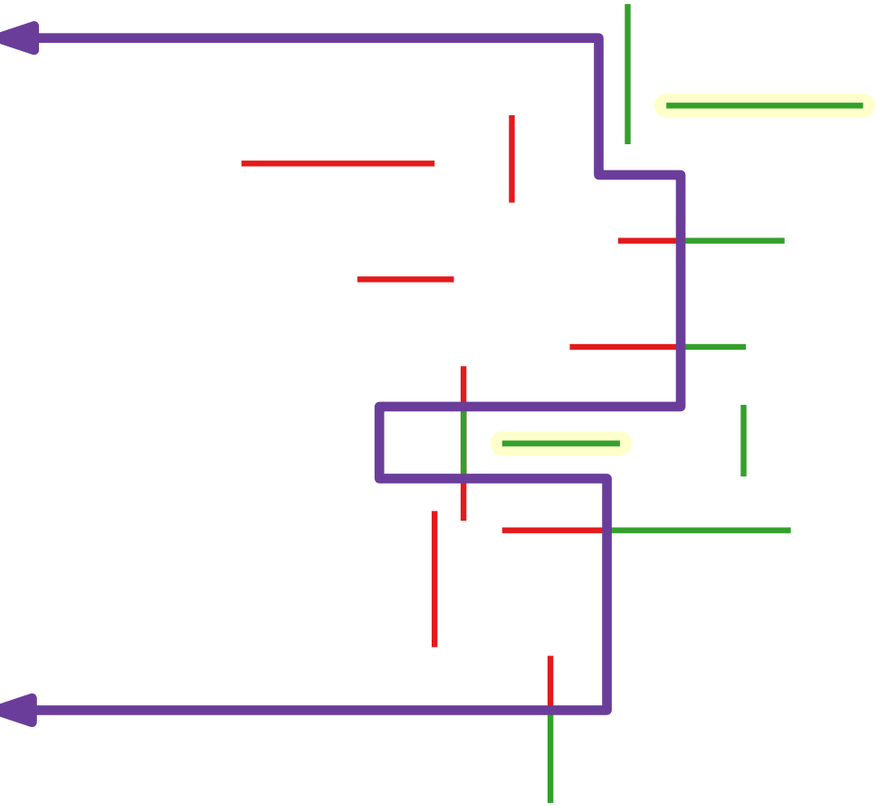


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F

Grenze und linkestes Label

MaxNumber 4S

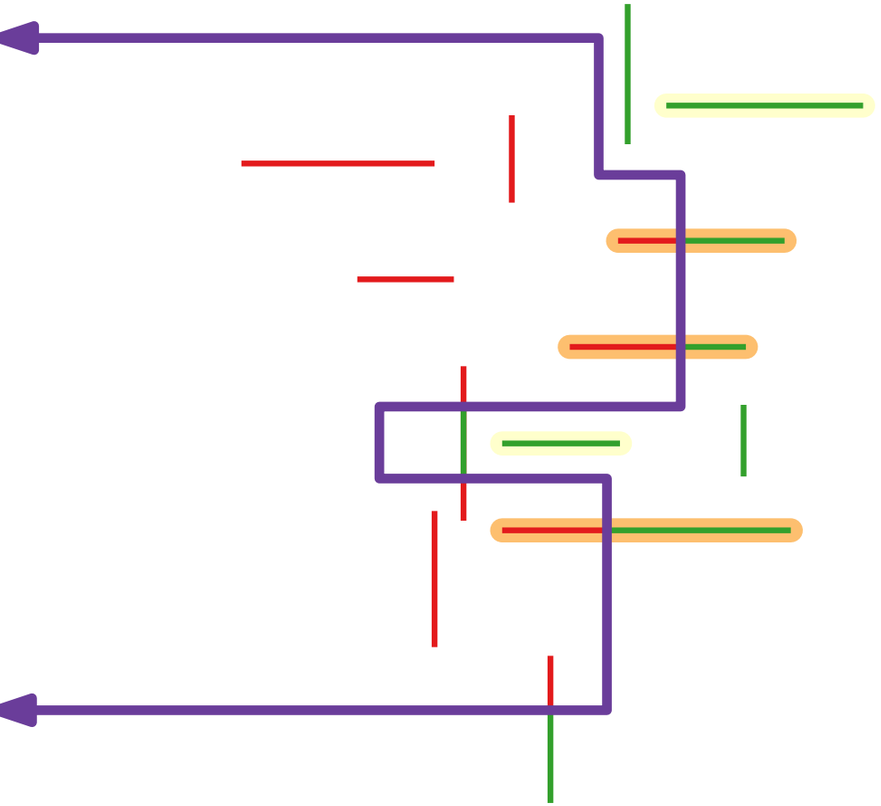
Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F

Grenze und linkestes Label

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

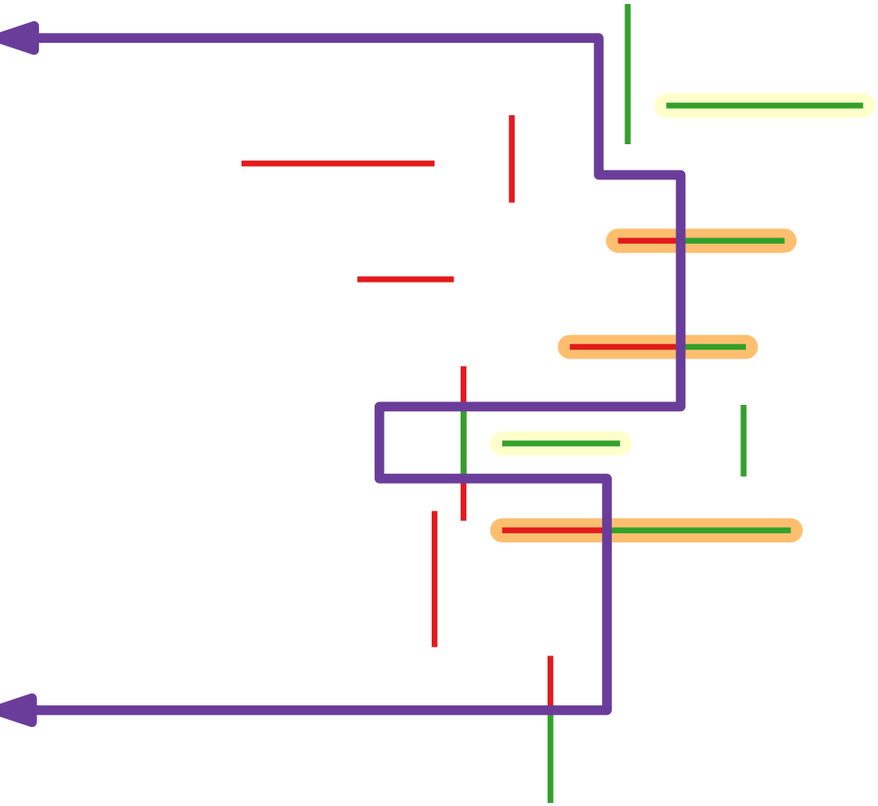


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F

Grenze und linkestes Label

MaxNumber 4S

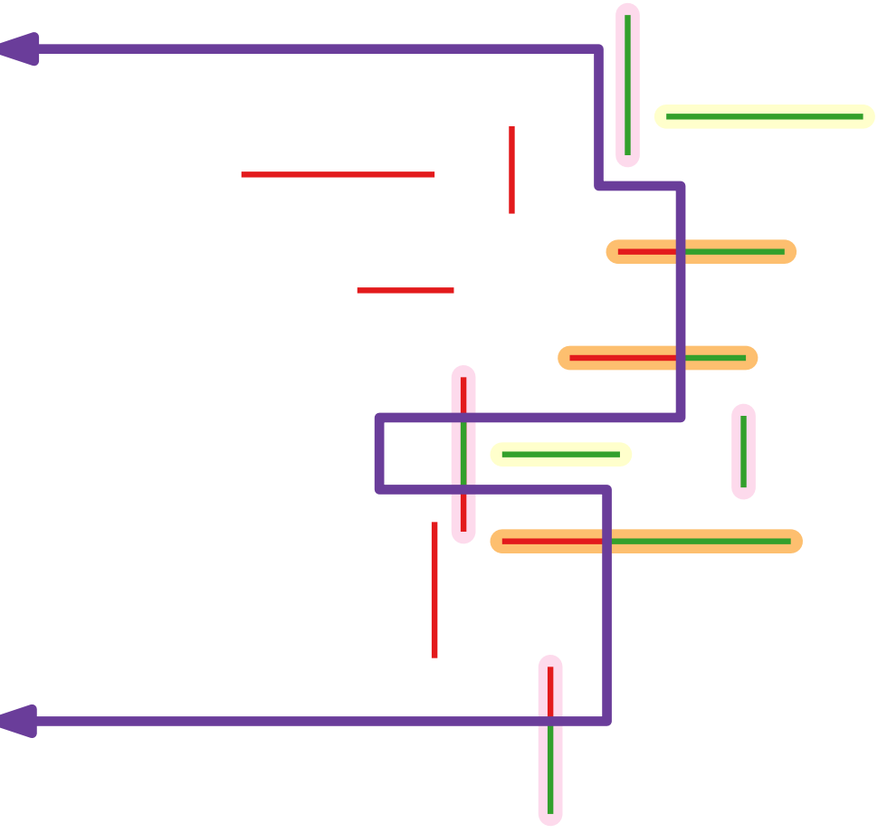
Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F

Grenze und linkestes Label

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.

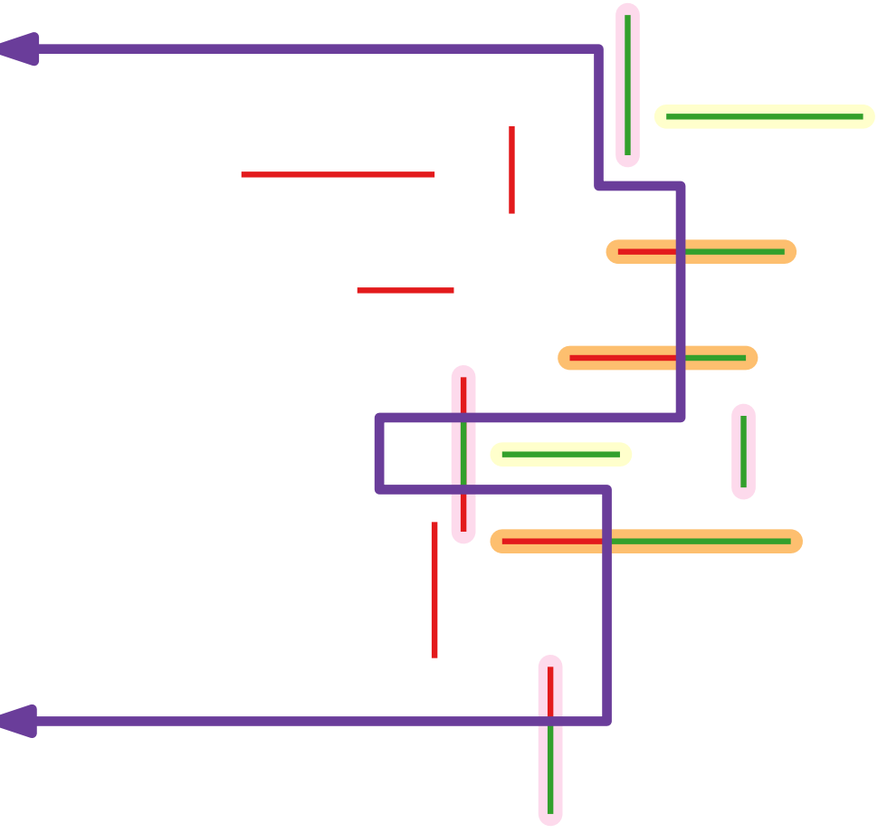


- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F

Grenze und linkestes Label

MaxNumber 4S

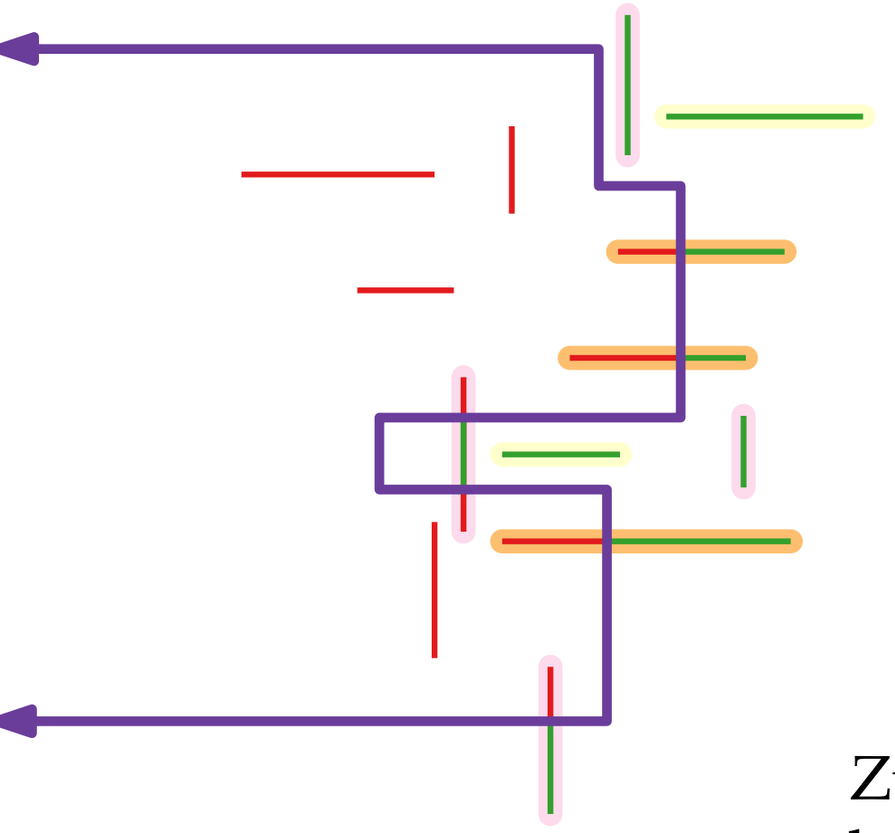
Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F
- Linkeste Punkte im grünen Bereich der Strecken sind zulässige Kandidaten

Grenze und linkestes Label

Zur Bestimmung des nächsten linkesten Labels genügt es, F und die Strecken $h_{2i-1}, h_{2i}, v_{2i-1}, v_{2i}$ für alle Punkte p_i rechts von F zu betrachten.



- Menge H der horizontalen Strecken
- Menge V der vertikalen Strecken
- $H_{\text{right}} \subseteq H$: komplett rechts von F
- $H_{\text{int}} \subseteq H$: schneiden F
- $V_{\text{int,right}} \subseteq V$: enthalten Punkt rechts von F
- Linkeste Punkte im grünen Bereich der Strecken sind zulässige Kandidaten

Zur effizienten Verwaltung der linkesten Kandidaten benötigen wir mehrere geometrische Datenstrukturen.

Datenstrukturen

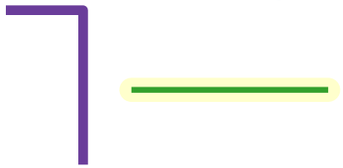
MaxNumber 4S

AGiS

Datenstrukturen

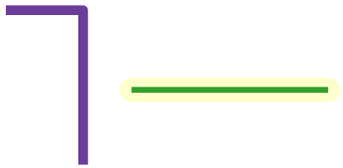
MaxNumber 4S

Menge H_{right}:



Datenstrukturen

Menge H_{right}:

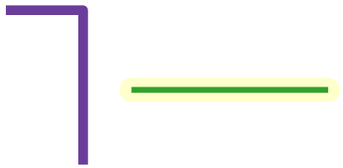


- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)

Datenstrukturen

MaxNumber 4S

Menge H_{right}:

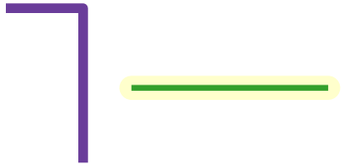


- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

Datenstrukturen

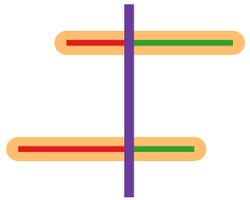
MaxNumber 4S

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

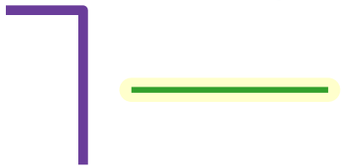
Menge H_{int}:



Datenstrukturen

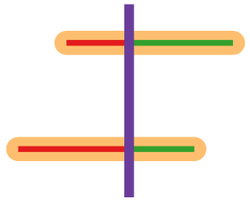
MaxNumber 4S

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

Menge H_{int}:

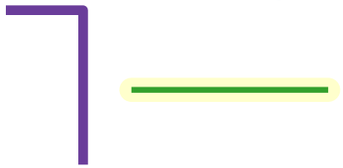


- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F

Datenstrukturen

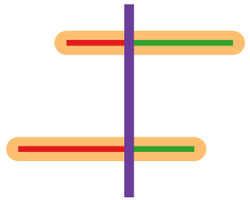
MaxNumber 4S

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

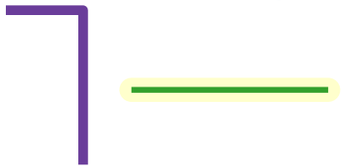
Menge H_{int}:



- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$

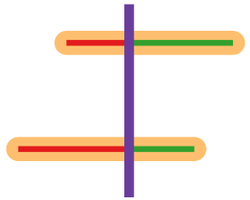
Datenstrukturen

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

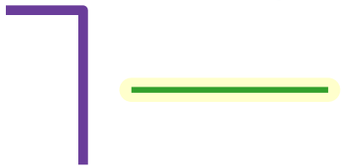
Menge H_{int}:



- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt

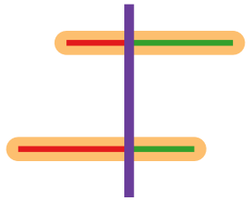
Datenstrukturen

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

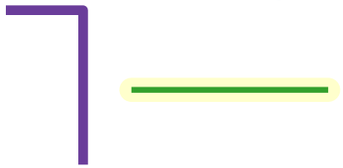
Menge H_{int}:



- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum

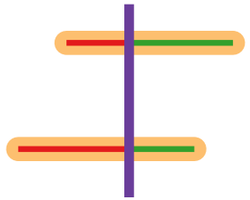
Datenstrukturen

Menge H_{right}:



- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

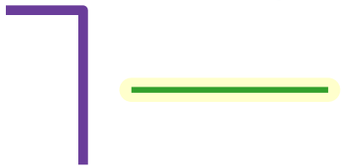
Menge H_{int}:



- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- MinHeap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

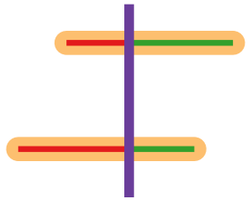
Datenstrukturen

Menge H_{right}:



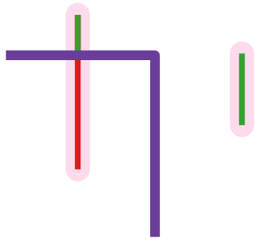
- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

Menge H_{int}:



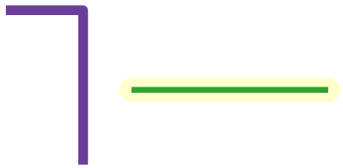
- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- MinHeap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

Menge $V_{\text{int,right}}$:



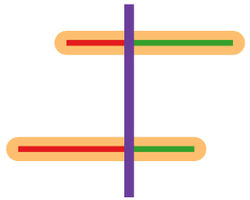
Datenstrukturen

Menge H_{right}:



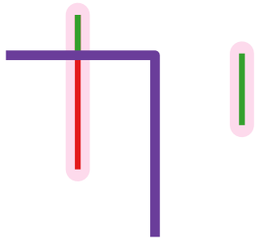
- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

Menge H_{int}:



- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- MinHeap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

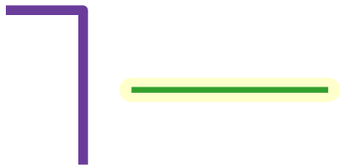
Menge $V_{\text{int,right}}$:



- MinHeap $\mathcal{H}_V$ geordnet nach x-Koordinaten der rechten Labelgrenzen

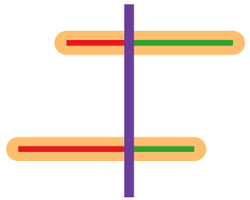
Datenstrukturen

Menge H_{right}:



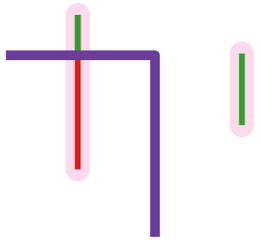
- speichere für jede Strecke in H_{right} deren rechten Endpunkt (= linkest mögliche Position der rechten Labelseite)
- Datenstruktur: MinHeap $\mathcal{H}_{\text{right}}$

Menge H_{int}:



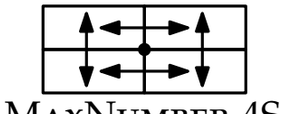
- Balanc. bin. Suchbaum $\mathcal{T}_i$ für jedes vertikale Segment f_i von F
- speichere Strecken aus H_{int}, die f_i schneiden sortiert nach y-Koordinaten in den Blättern von $\mathcal{T}_i$
- speichere zusätzlich Labelbreite an jedem Blatt
- an inneren Knoten minimale Labelbreite im Teilbaum
- MinHeap $\mathcal{H}_{\text{int}}$ für linkestes Label in jedem $\mathcal{T}_i$

Menge $V_{\text{int,right}}$:



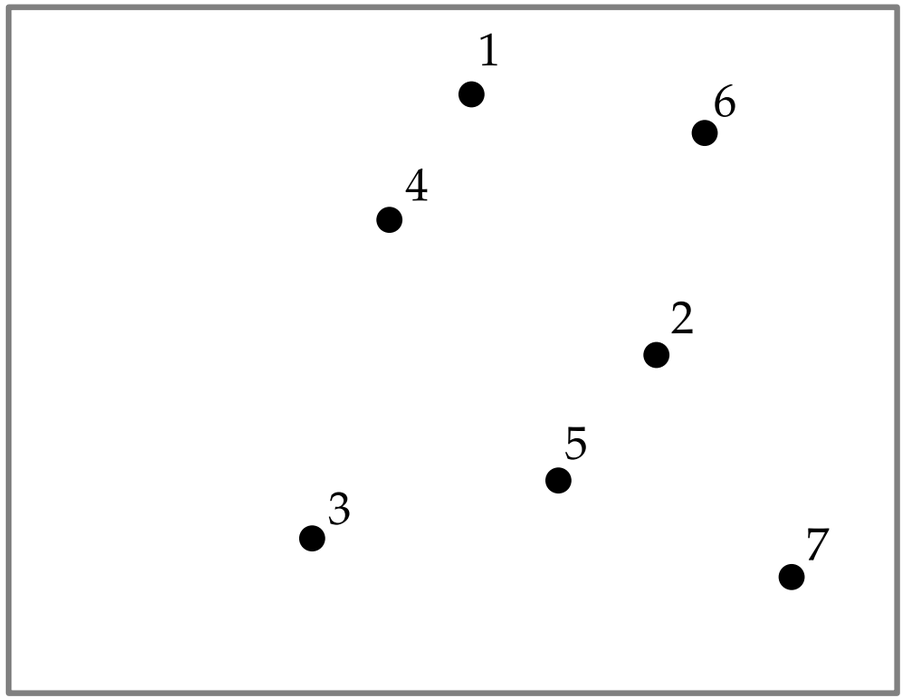
- MinHeap $\mathcal{H}_V$ geordnet nach x-Koordinaten der rechten Labelgrenzen
- Lazy Strategie: Segment wird erst aus $\mathcal{H}_V$ geworfen, wenn es linkestes Label liefern würde, aber nicht mehr in $V_{\text{int,right}}$ liegt.

Priority Search Trees (PRT)



Priority Search Trees (PRT)

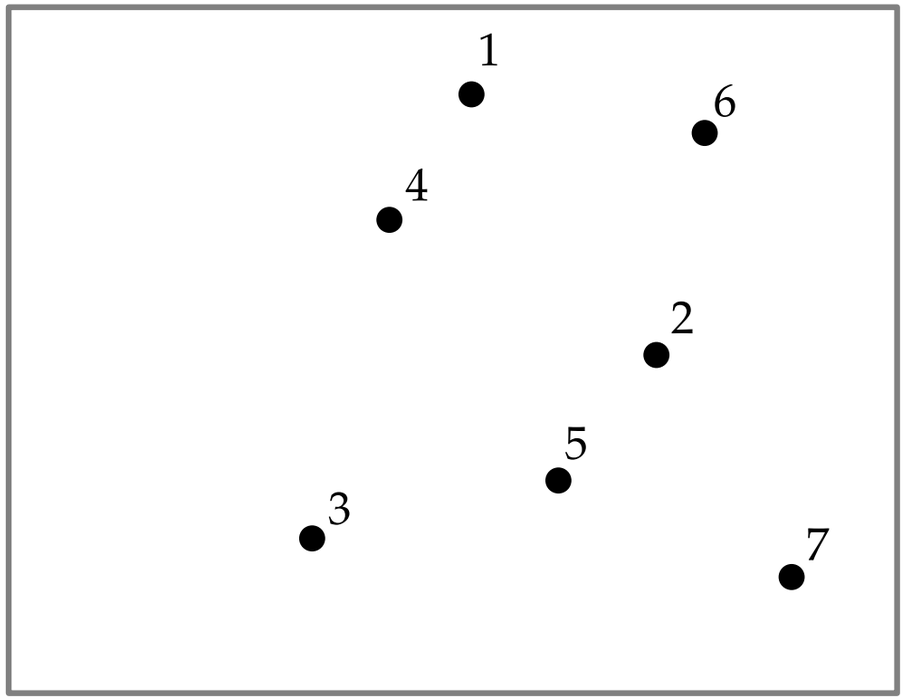
MaxNumber 4S



Priority Search Trees (PRT)

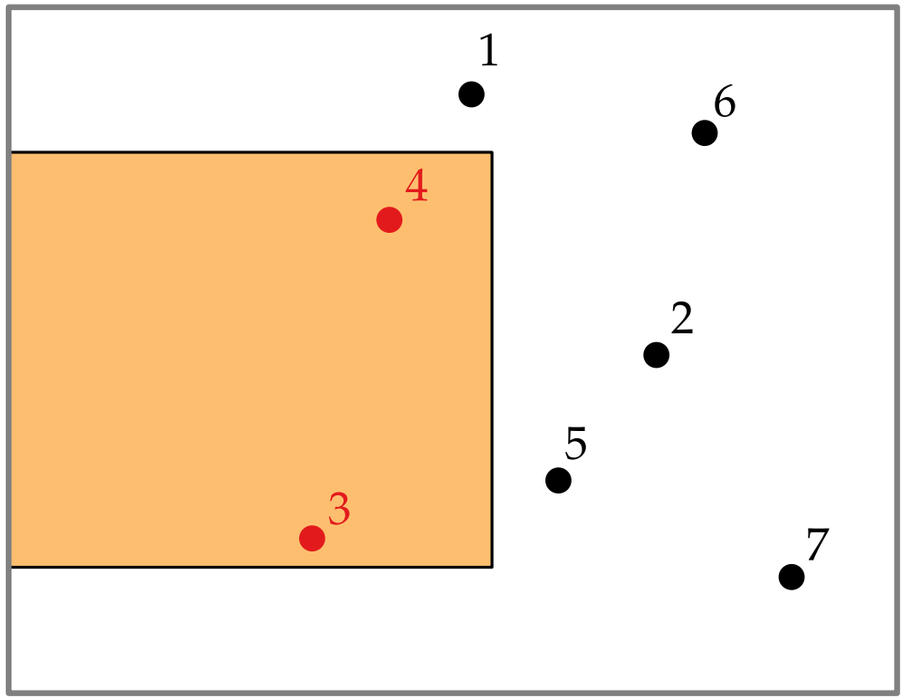
MaxNumber 4S

- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$



Priority Search Trees (PRT)

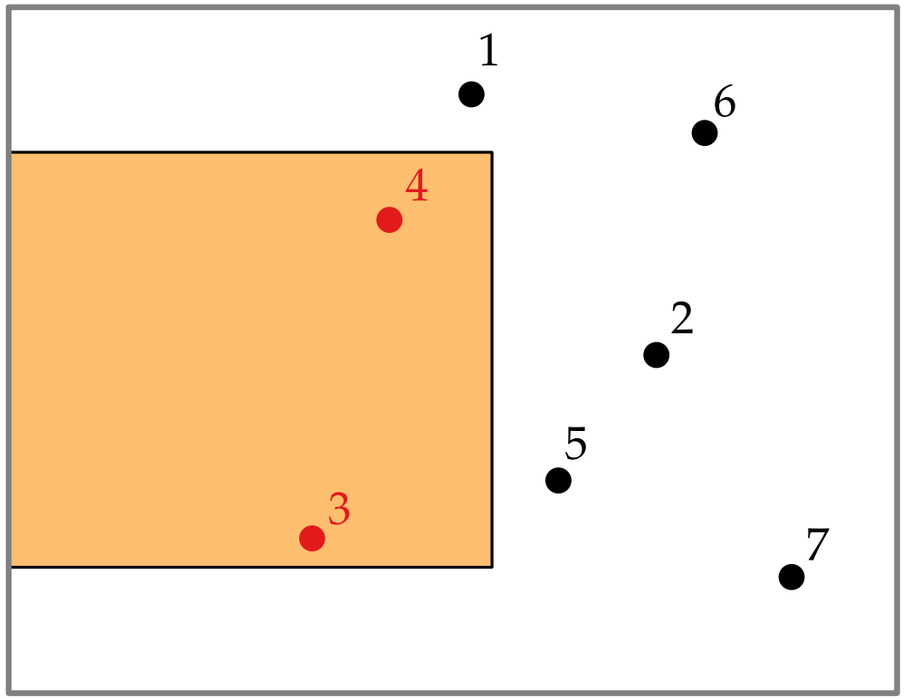
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$



Priority Search Trees (PRT)

MaxNumber 4S

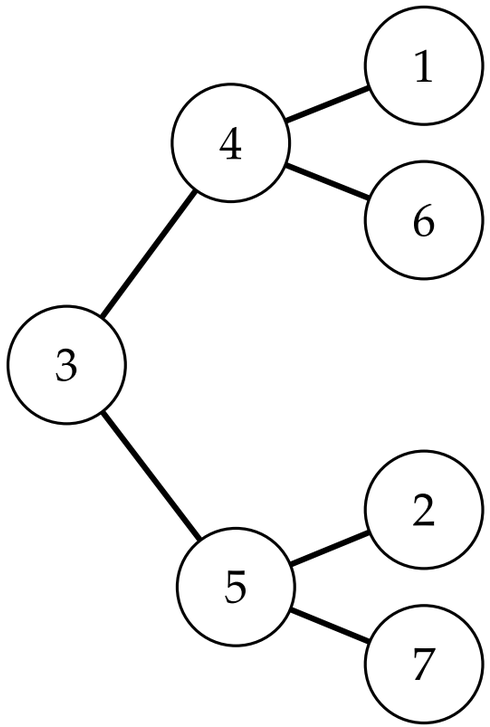
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x

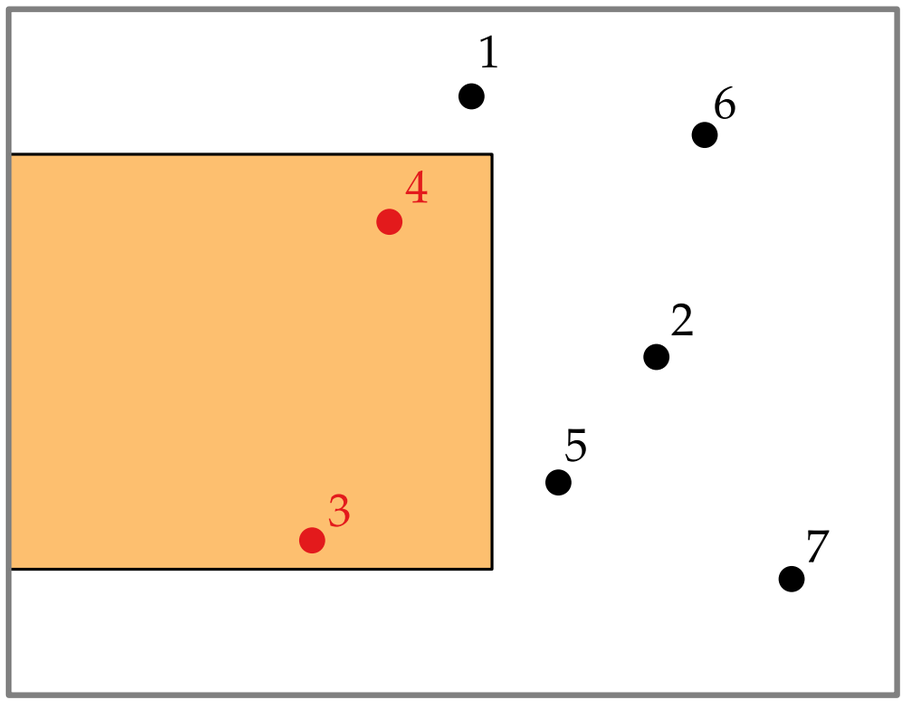


Priority Search Trees (PRT)

MaxNumber 4S

- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x

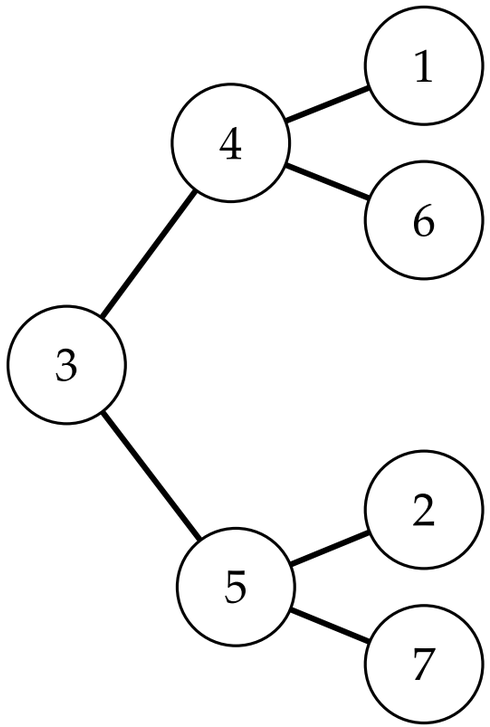


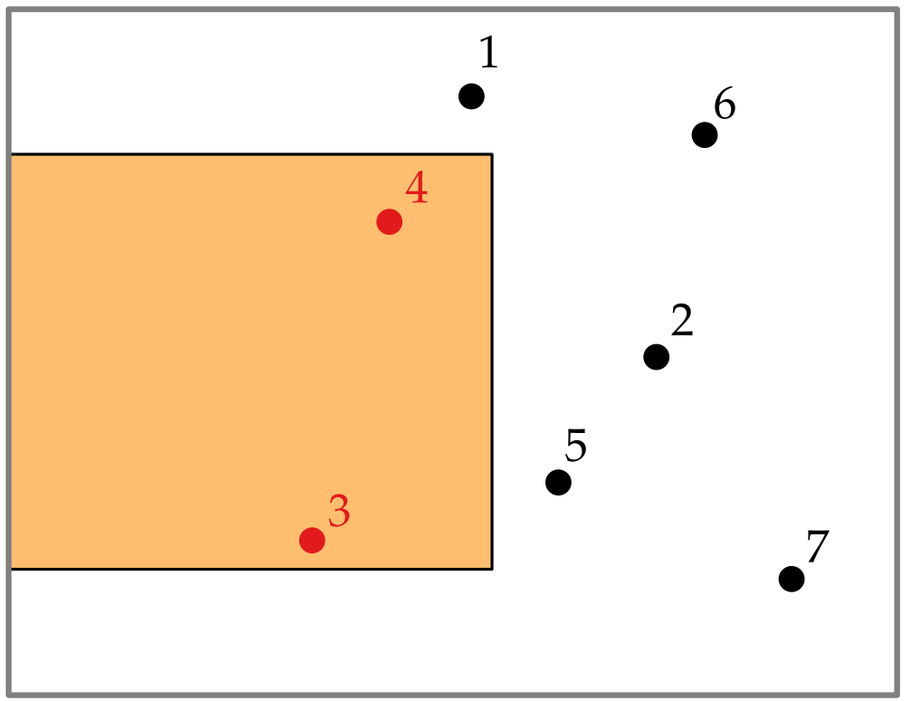


Priority Search Trees (PRT)

MaxNumber 4S

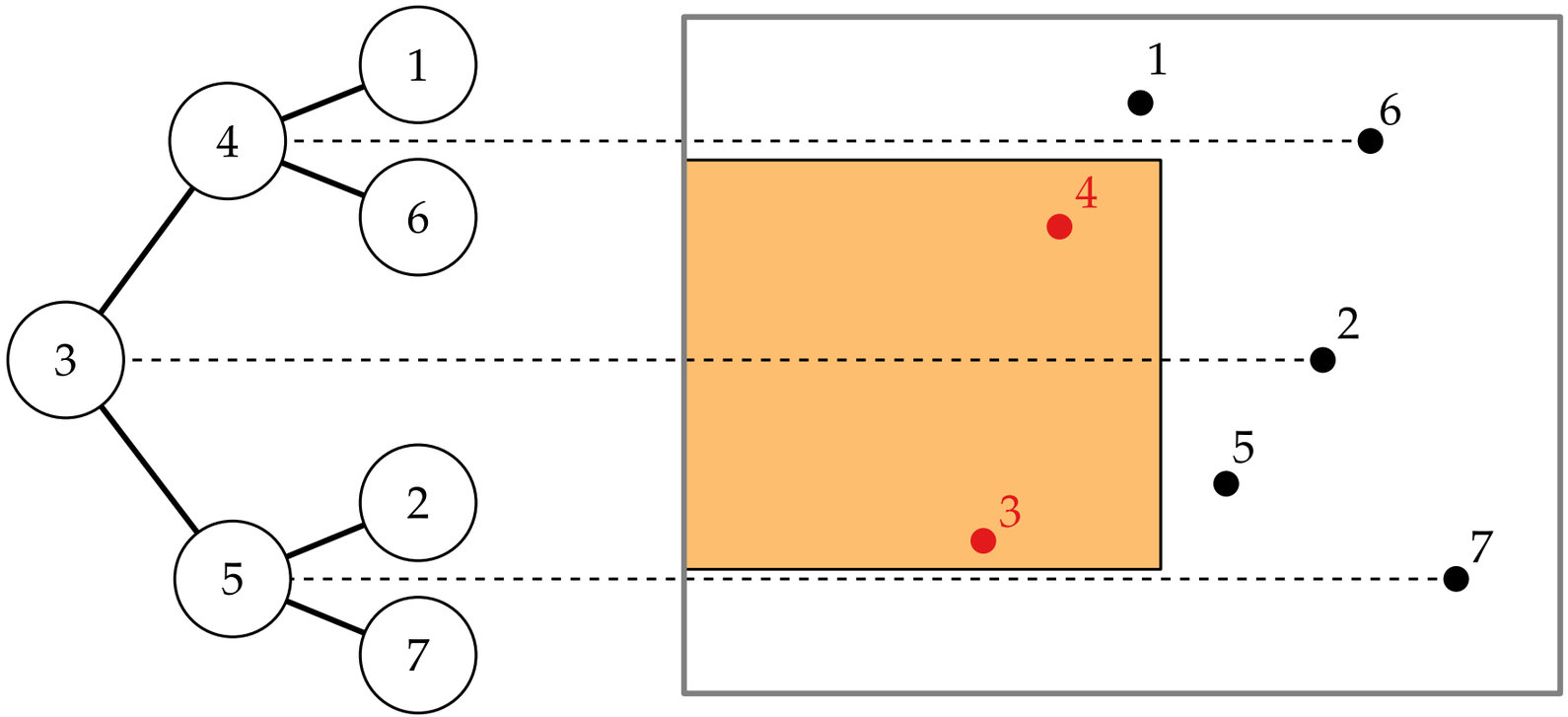
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y





Priority Search Trees (PRT)

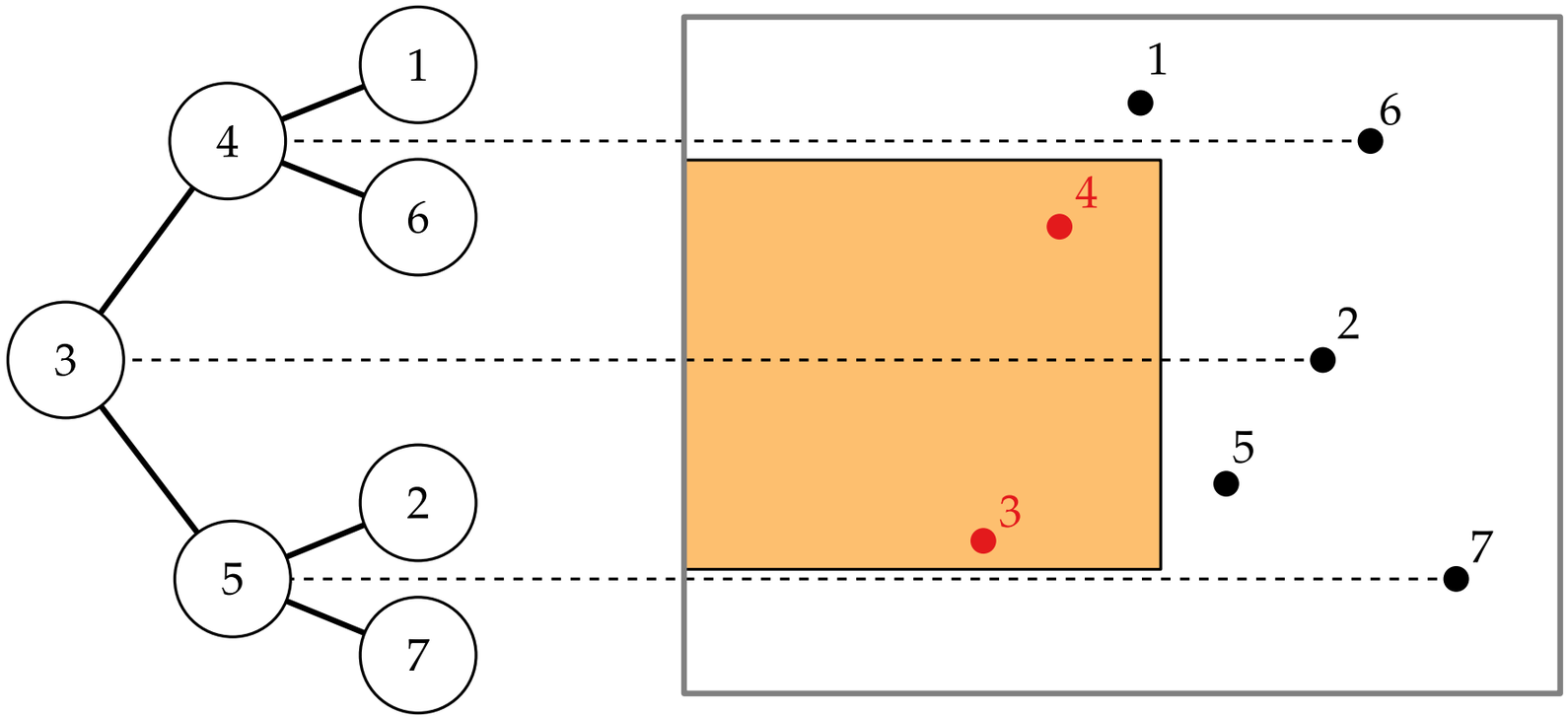
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y



Priority Search Trees (PRT)

MaxNumber 4S

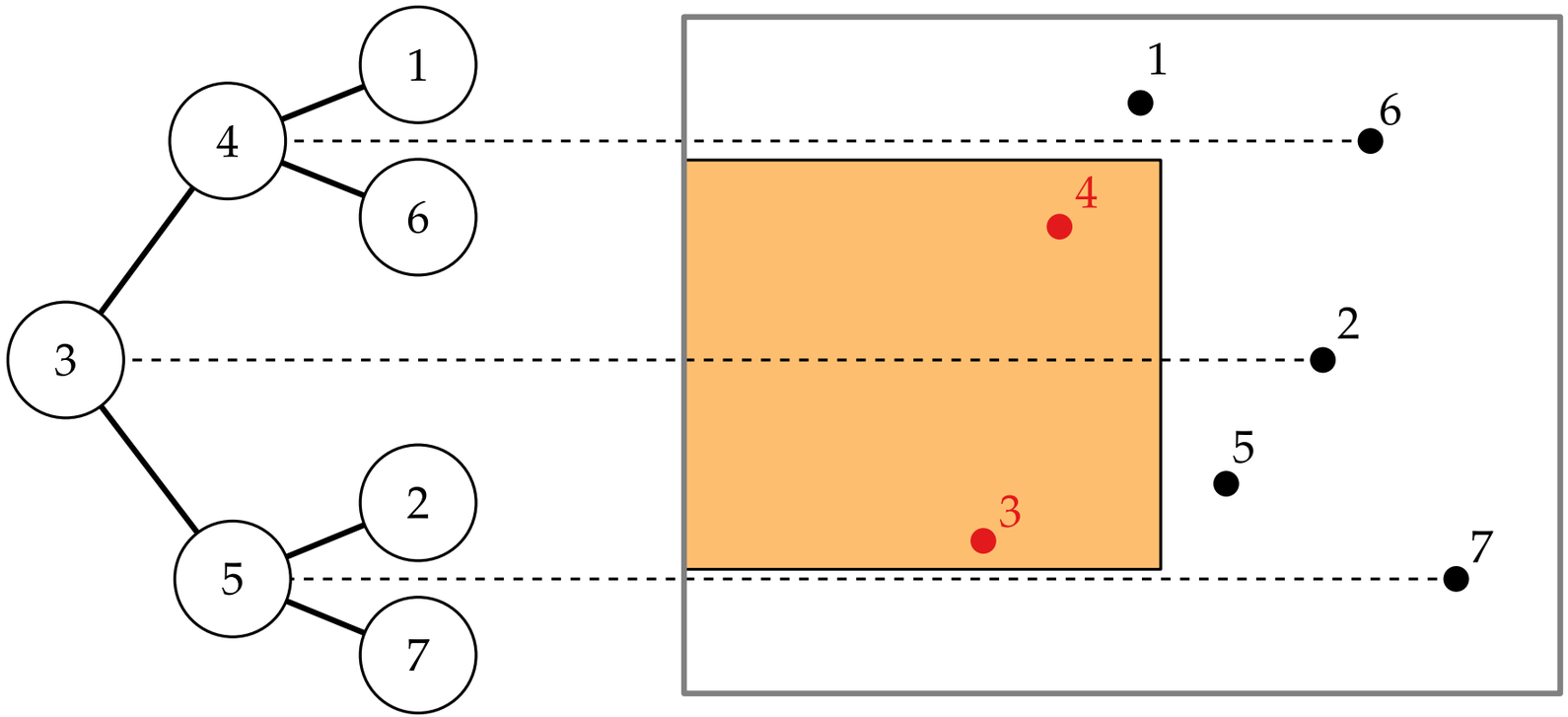
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$



Priority Search Trees (PRT)

MaxNumber 4S

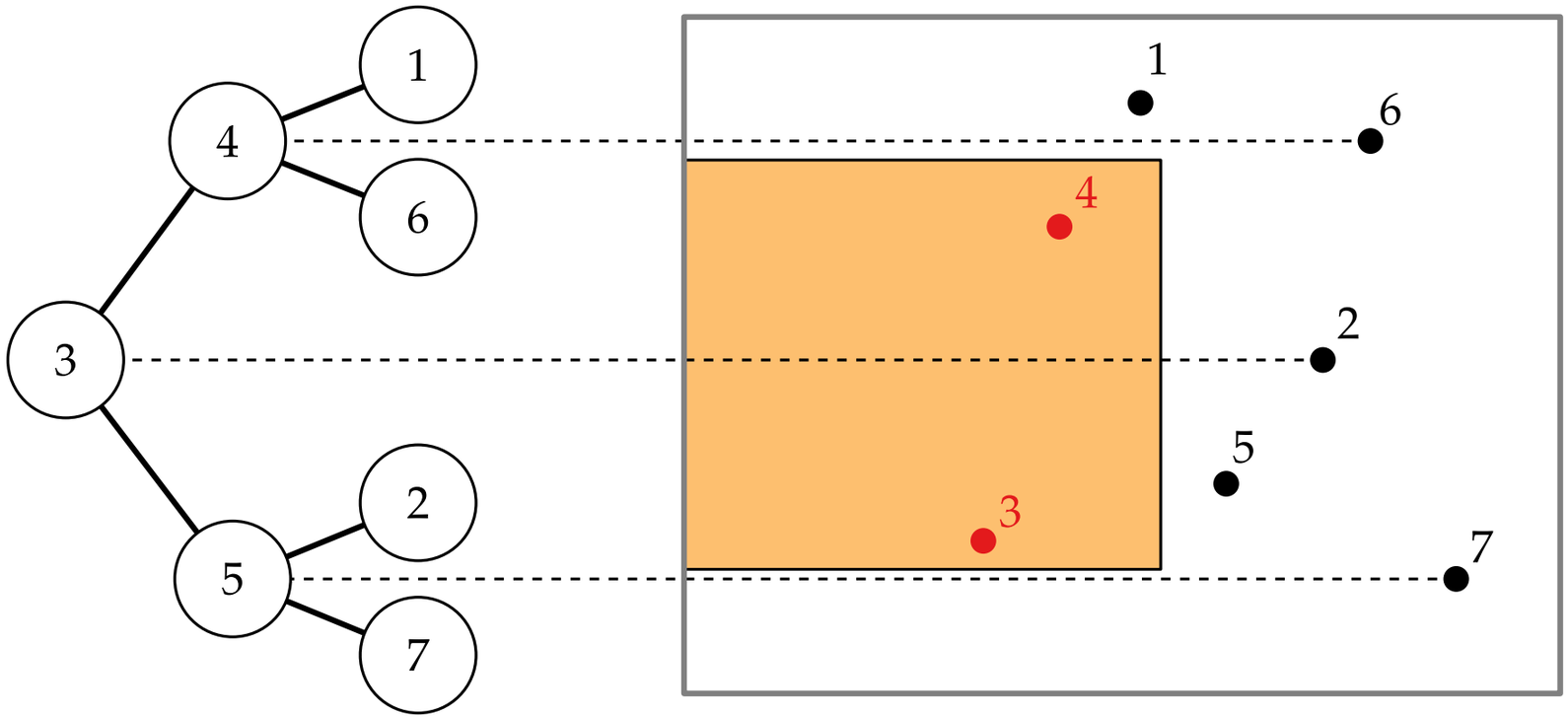
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit



Priority Search Trees (PRT)

MaxNumber 4S

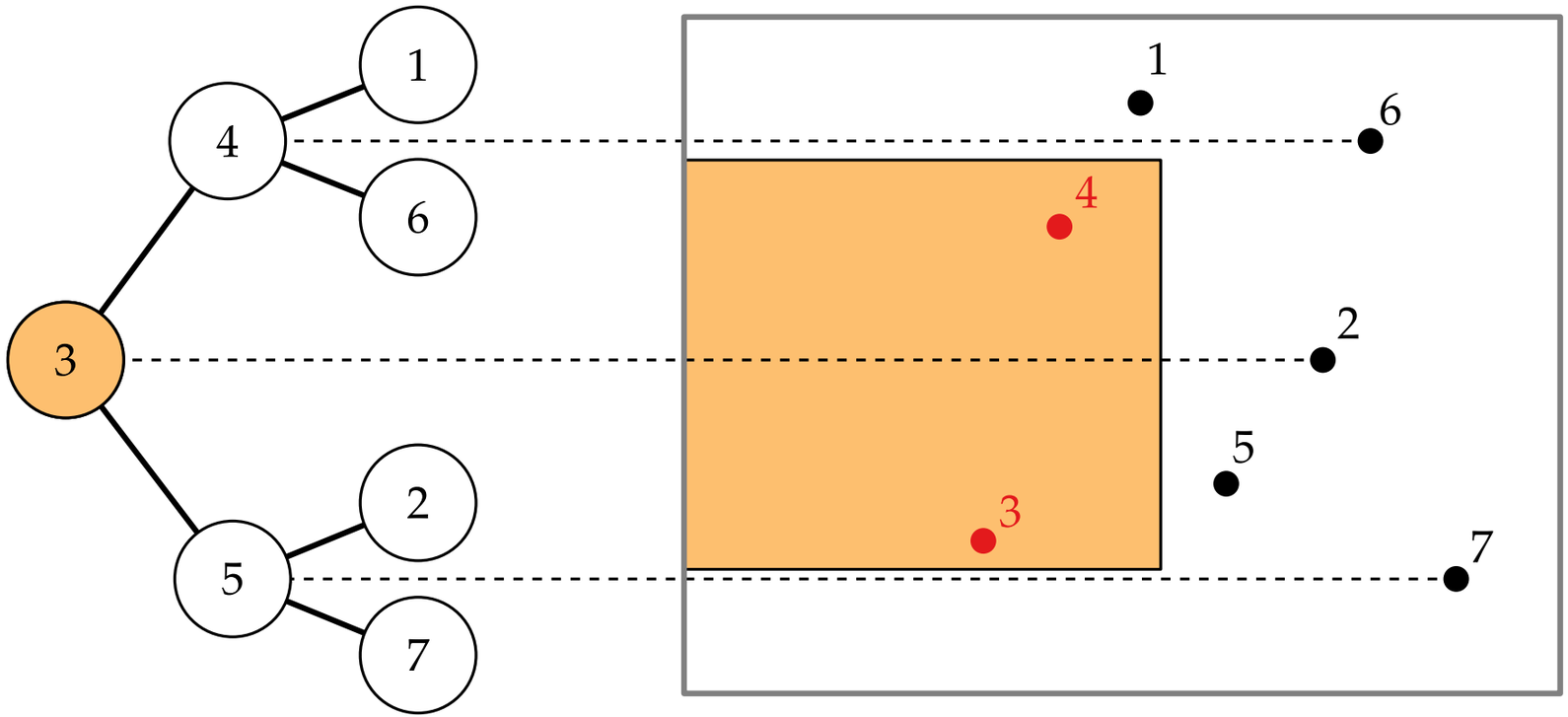
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

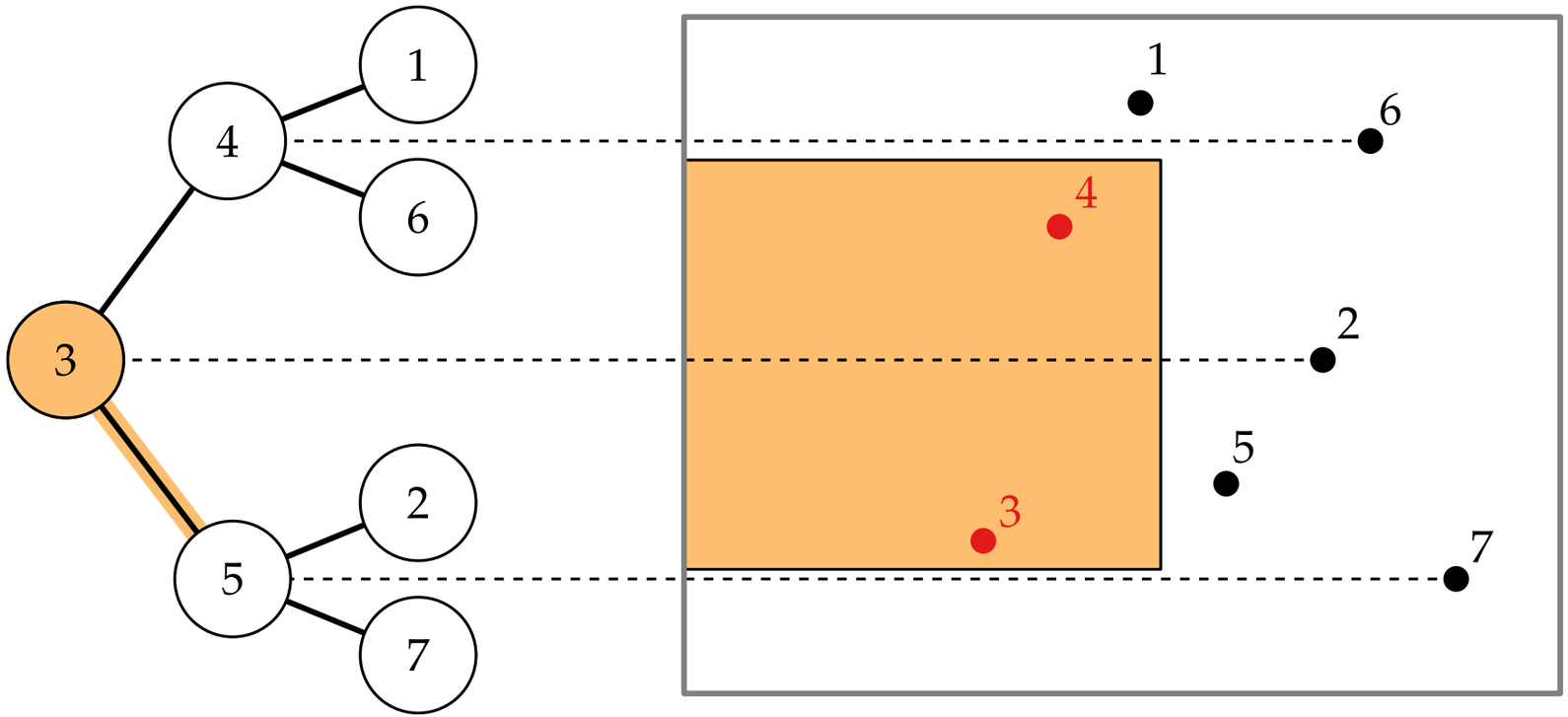
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

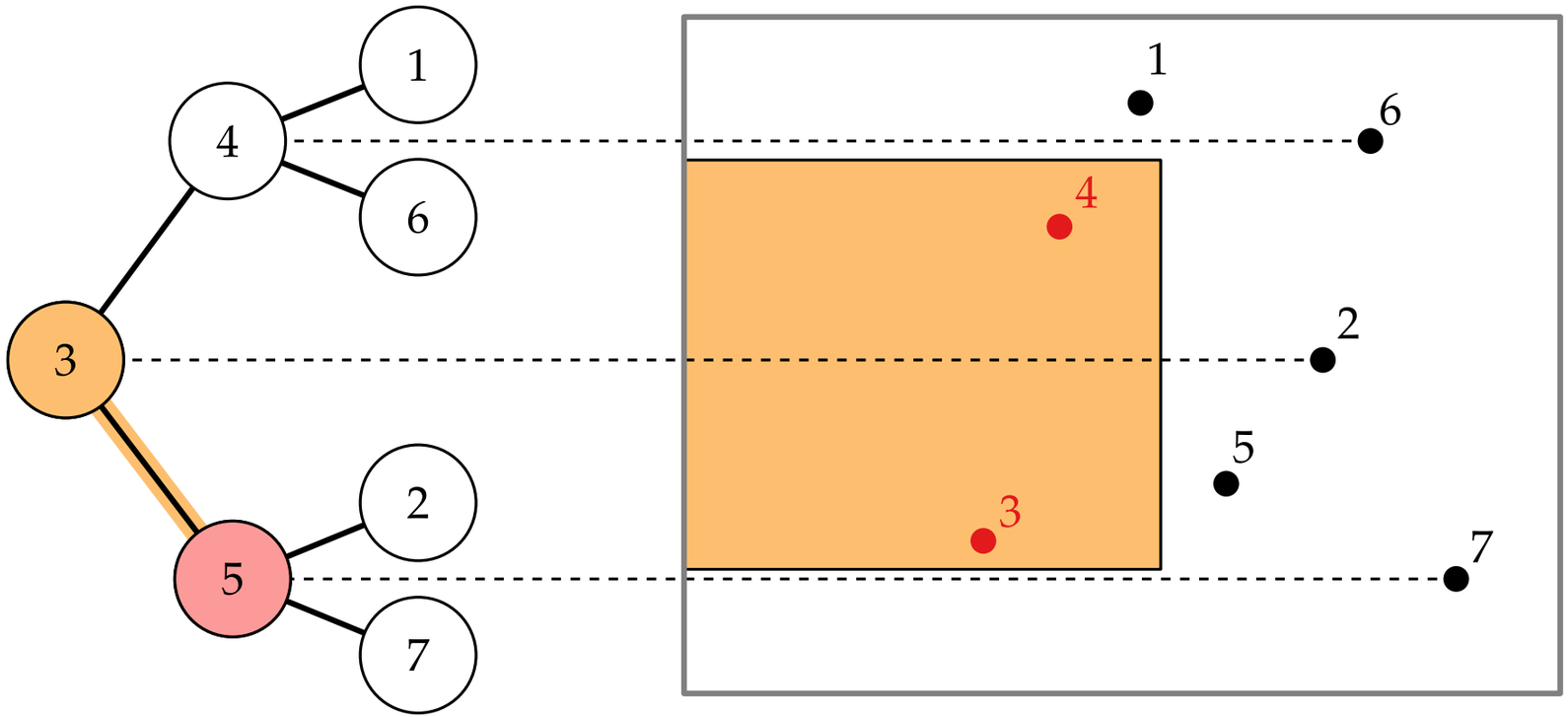
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

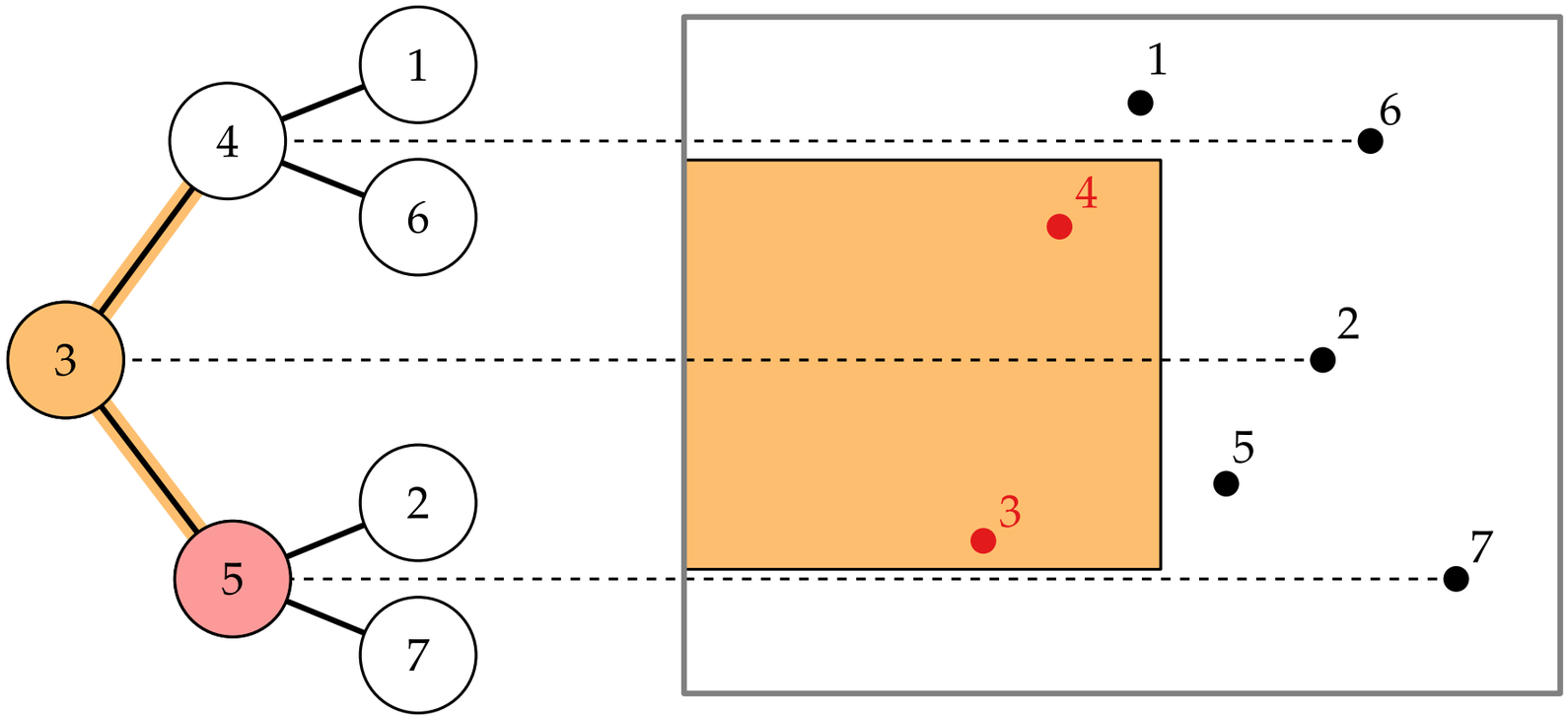
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

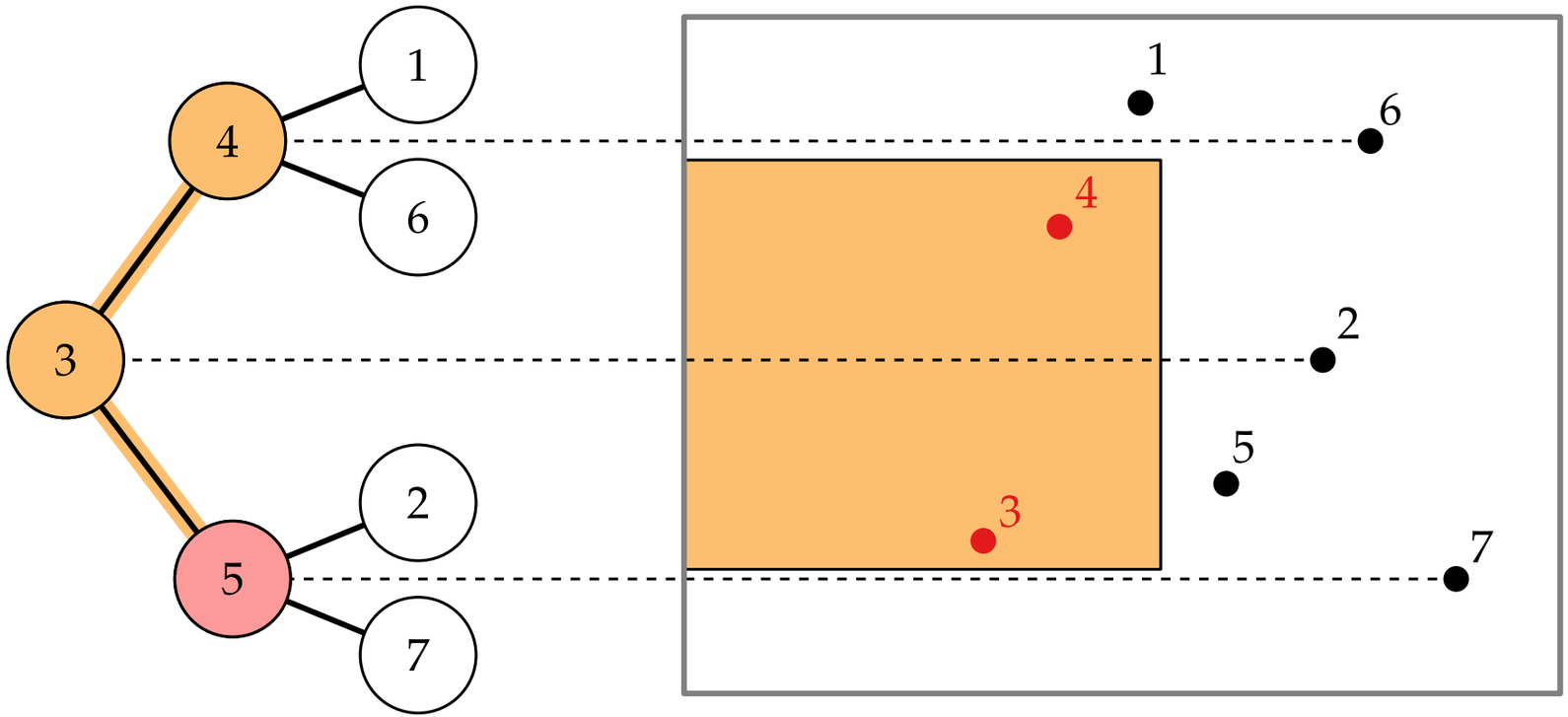
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

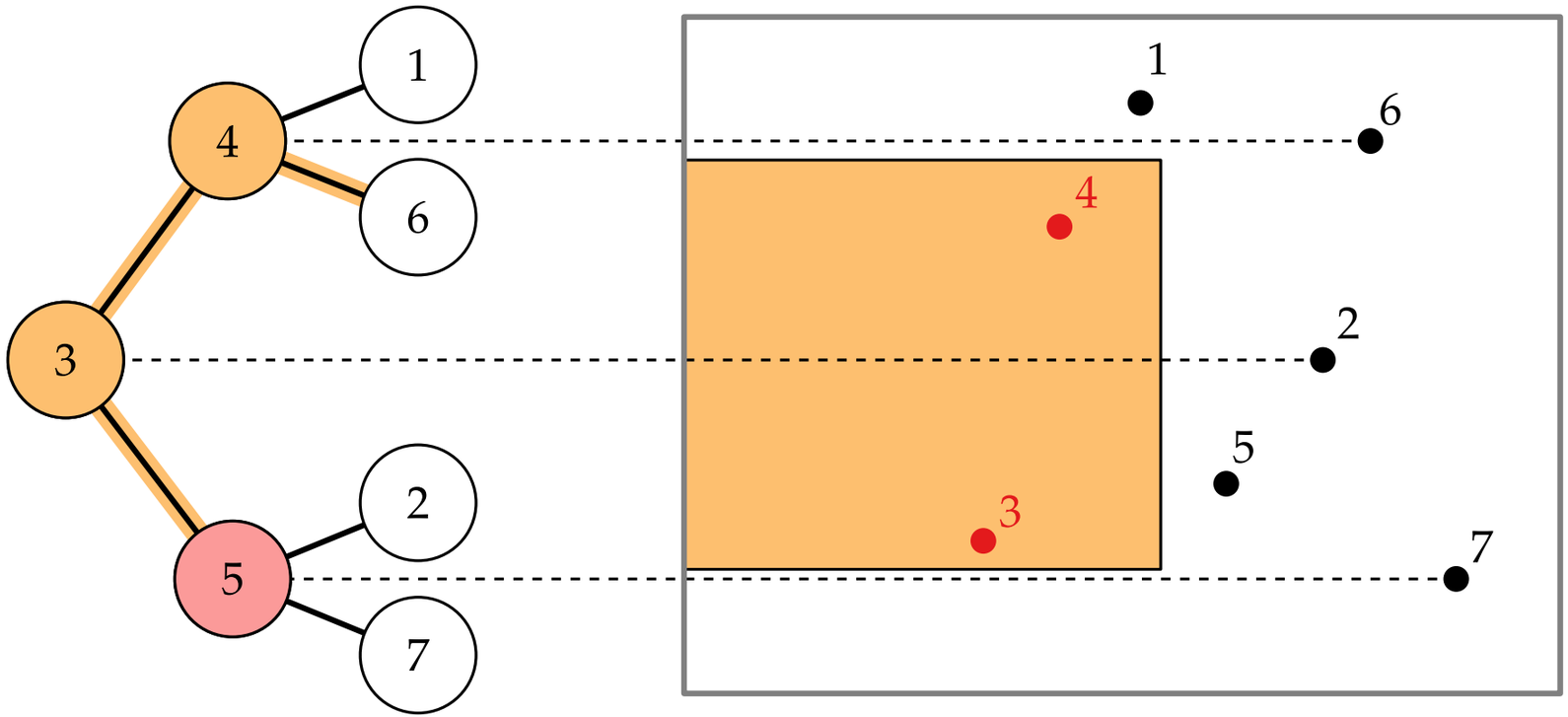
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



Priority Search Trees (PRT)

MaxNumber 4S

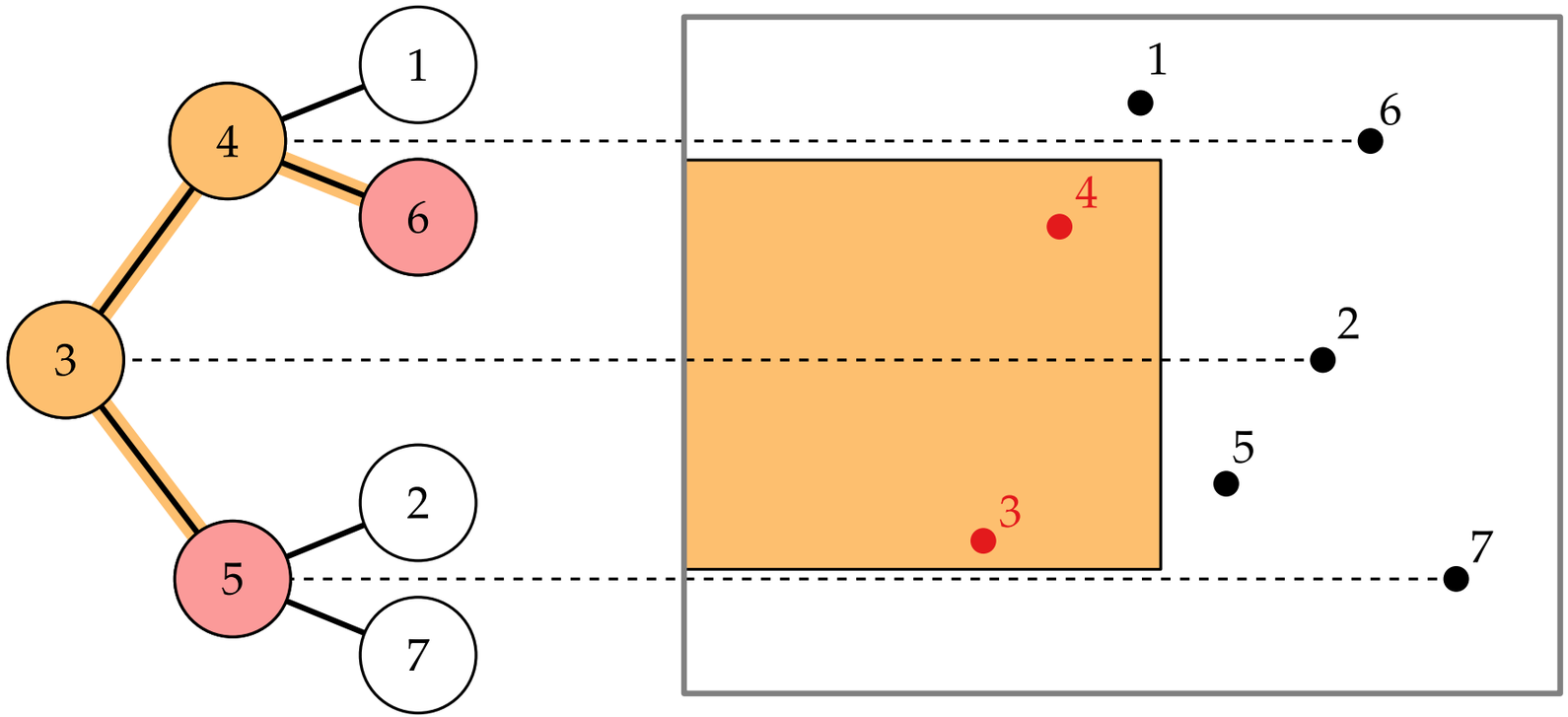
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



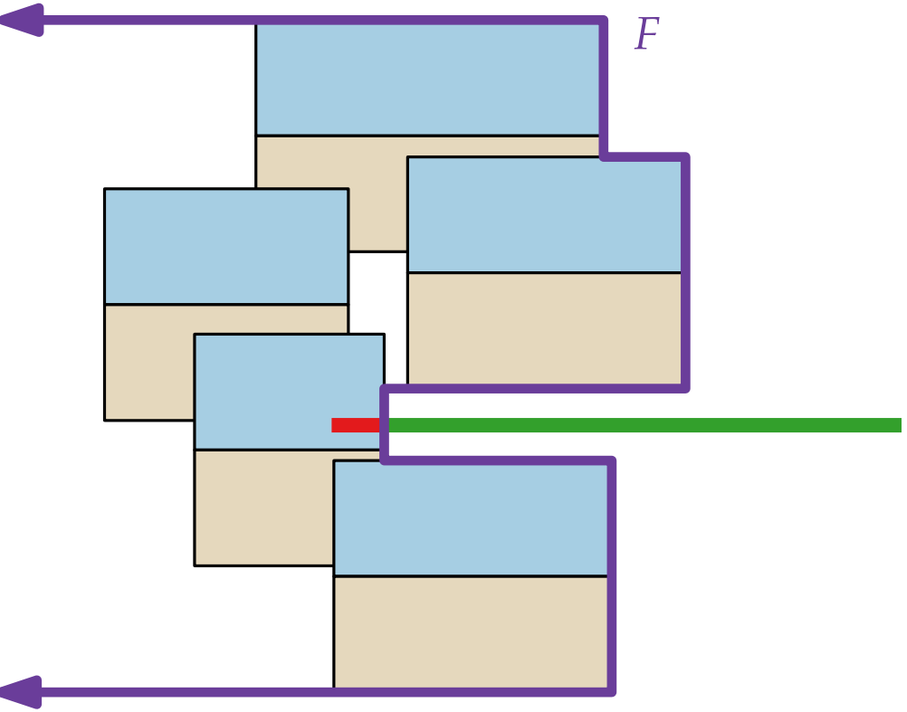
Priority Search Trees (PRT)

MaxNumber 4S

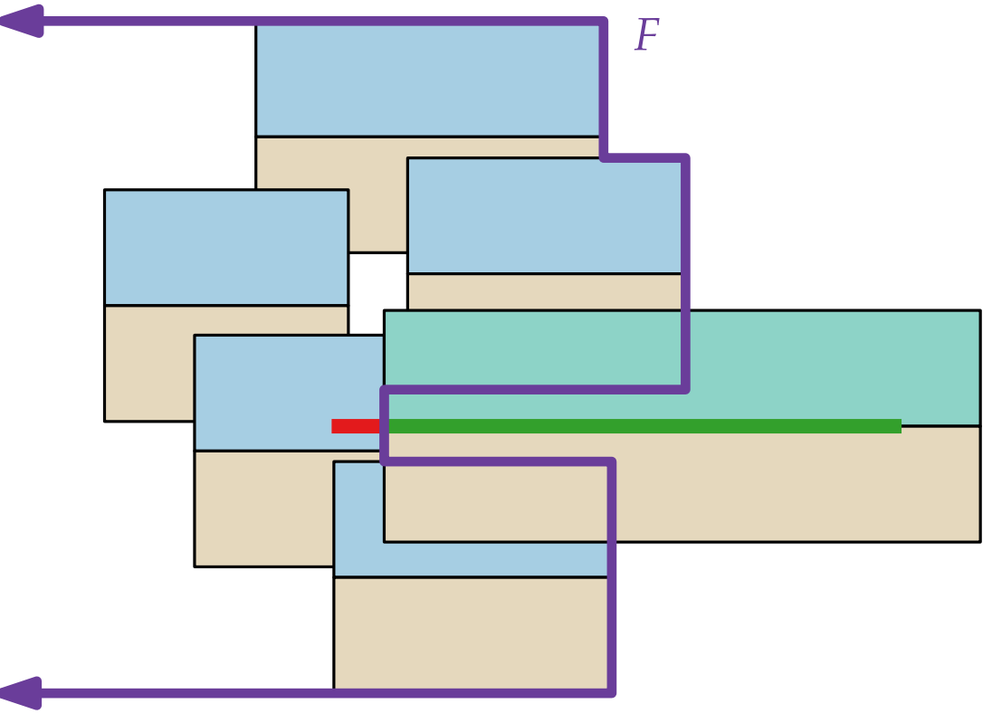
- Datenstruktur für Bereichsabfragen der Form $[-\infty, x] \times [y, y']$
- Heap-Eigenschaft für x
- Suchbaum für y
- Größe $\mathcal{O}(n)$
- Einfügen und Löschen in $\mathcal{O}(\log n)$ Zeit
- Suchen in $\mathcal{O}(k + \log n)$ Zeit (k =Ausgabegröße)



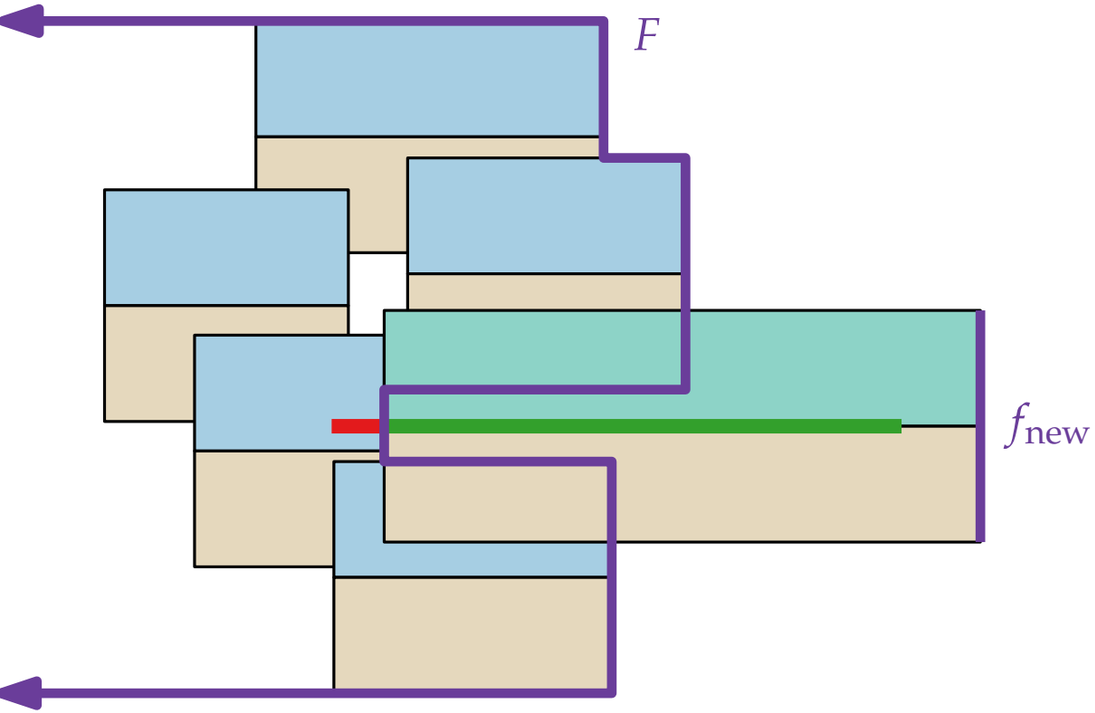
Updates



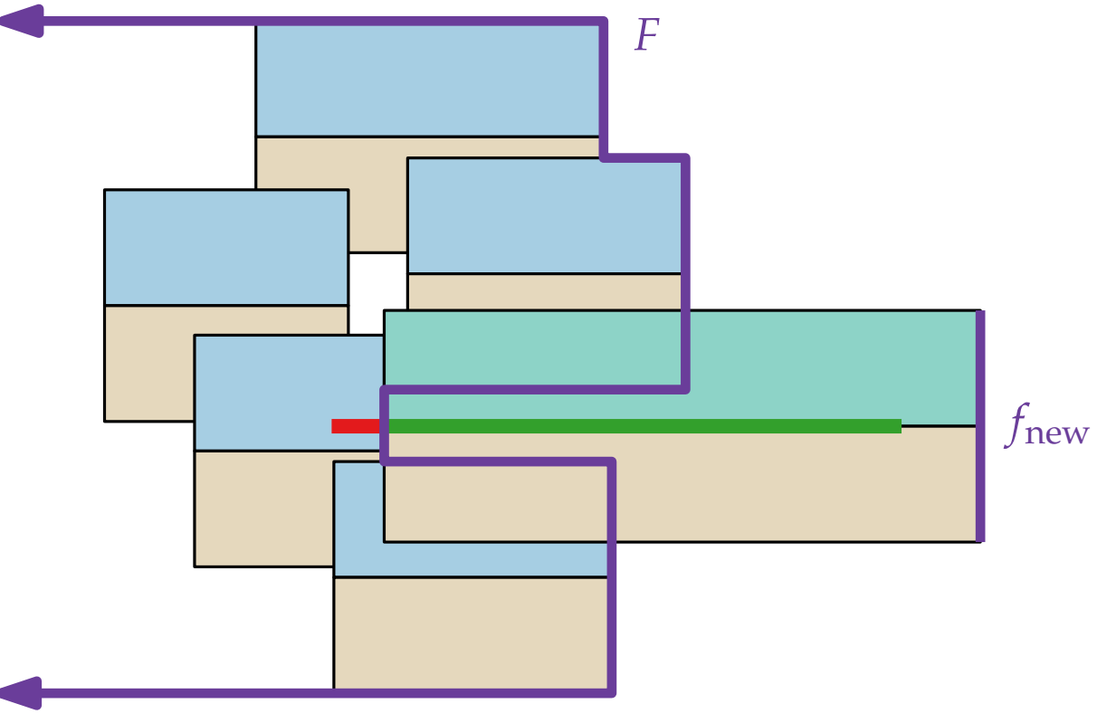
Updates



Updates

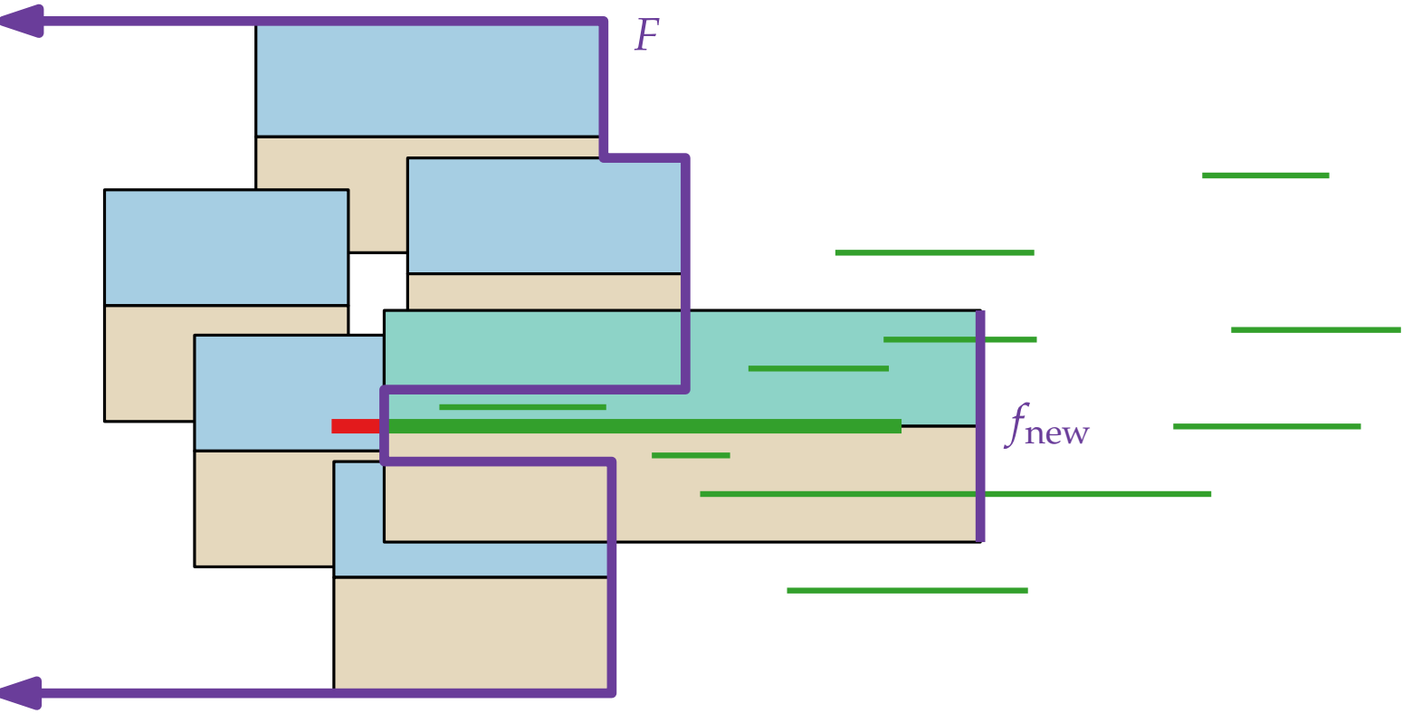


Updates



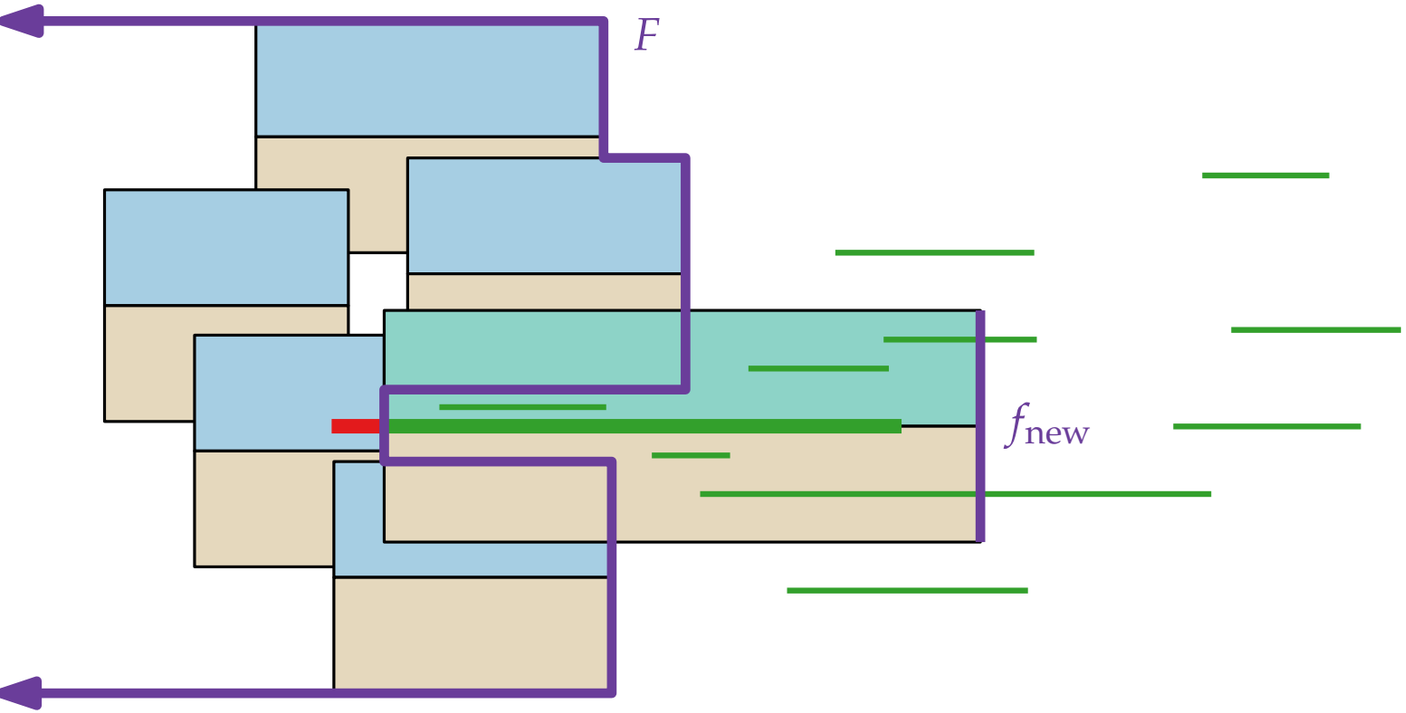
- Updates von H_{right}

Updates



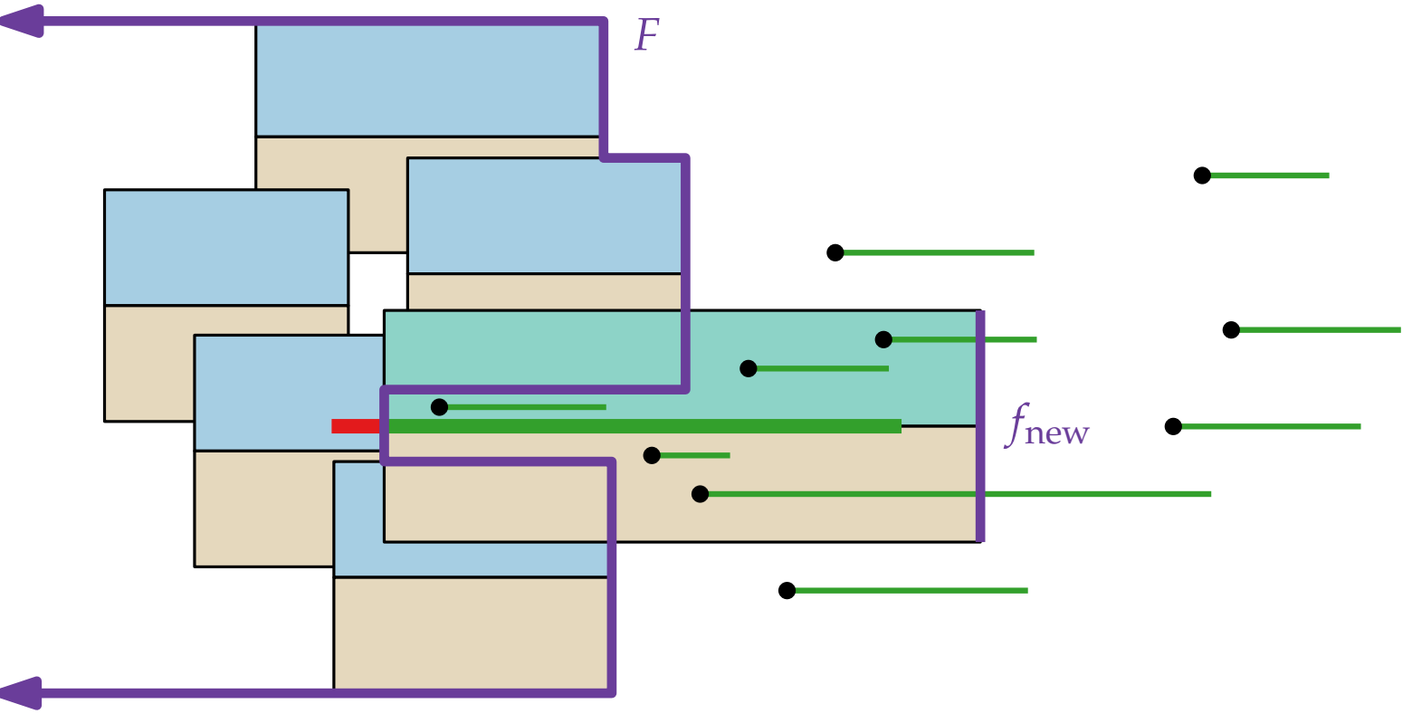
- Updates von H_{right}

Updates



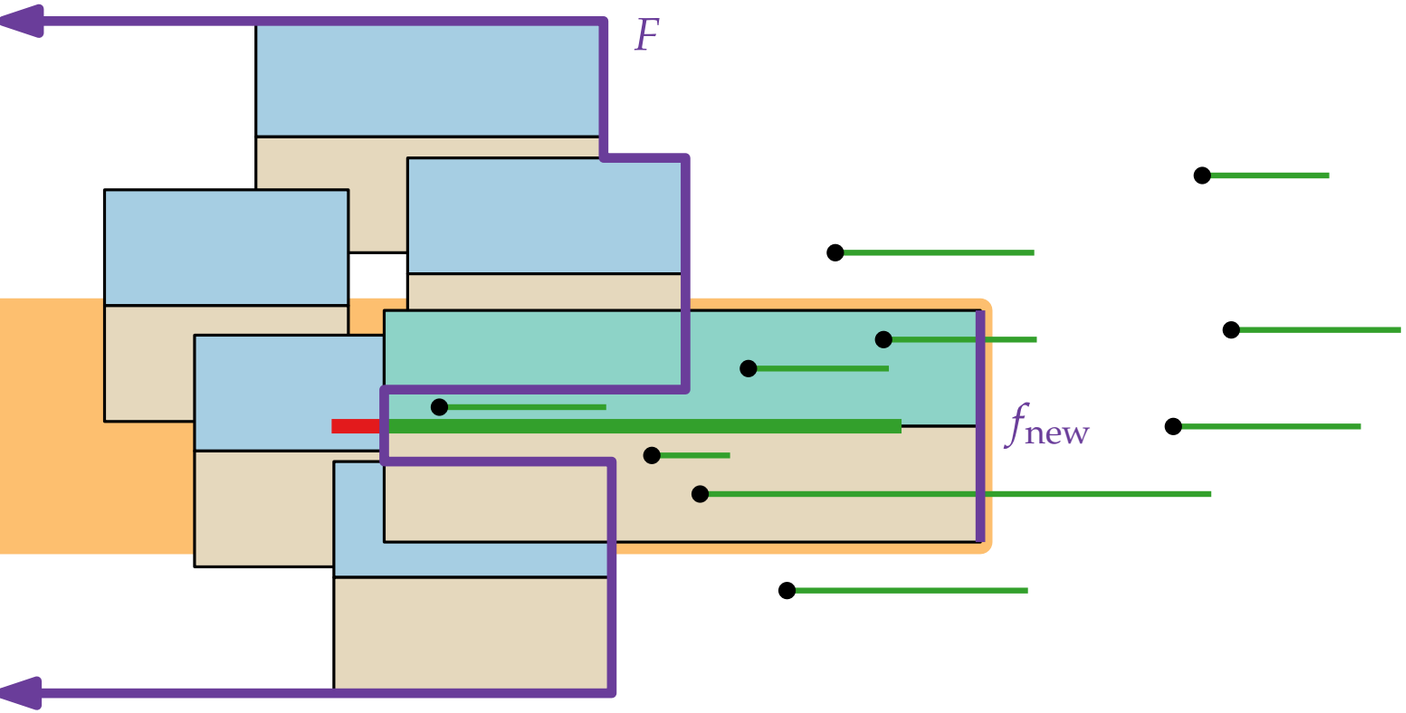
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken

Updates



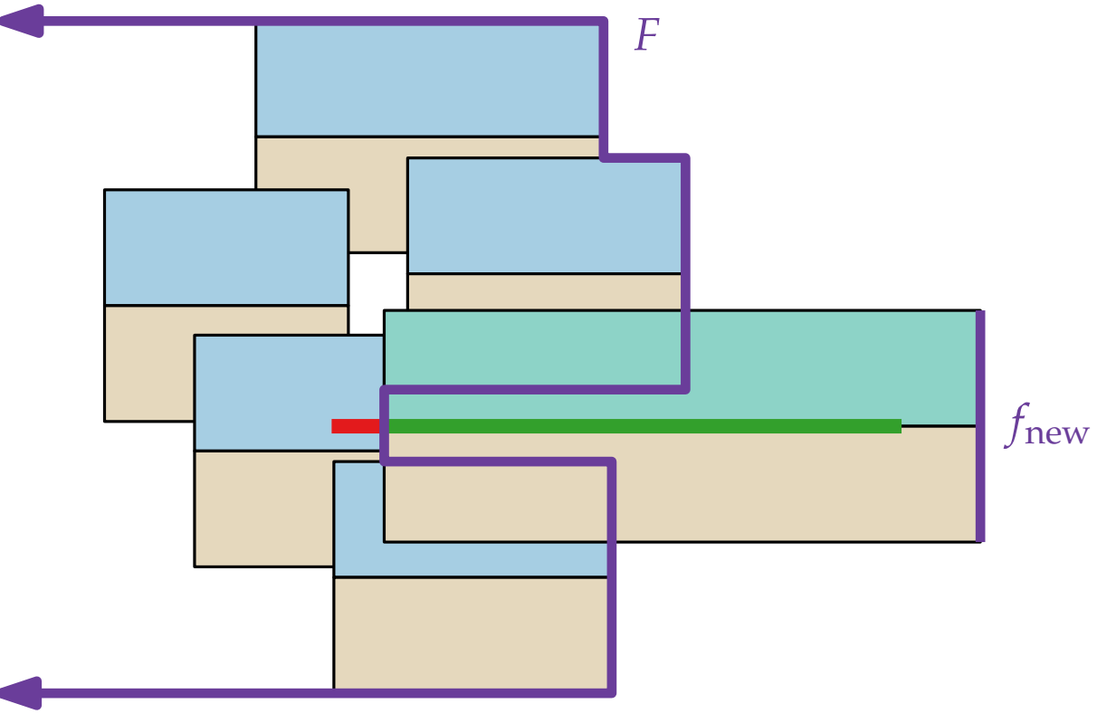
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken

Updates



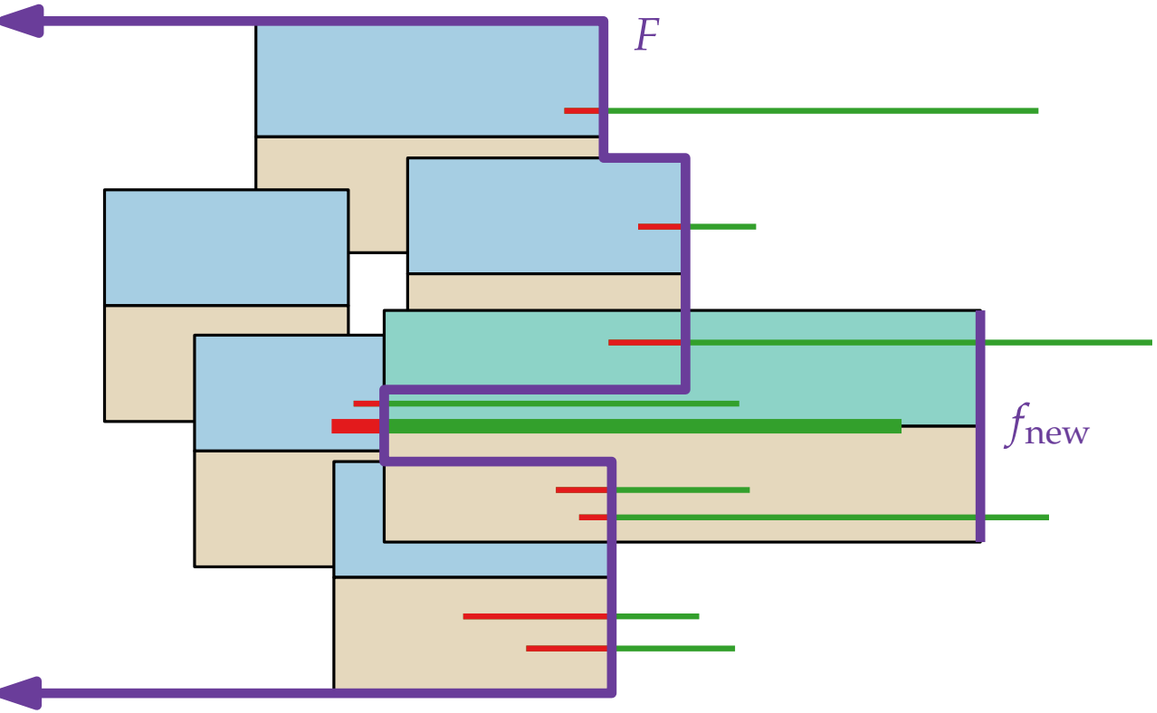
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken

Updates



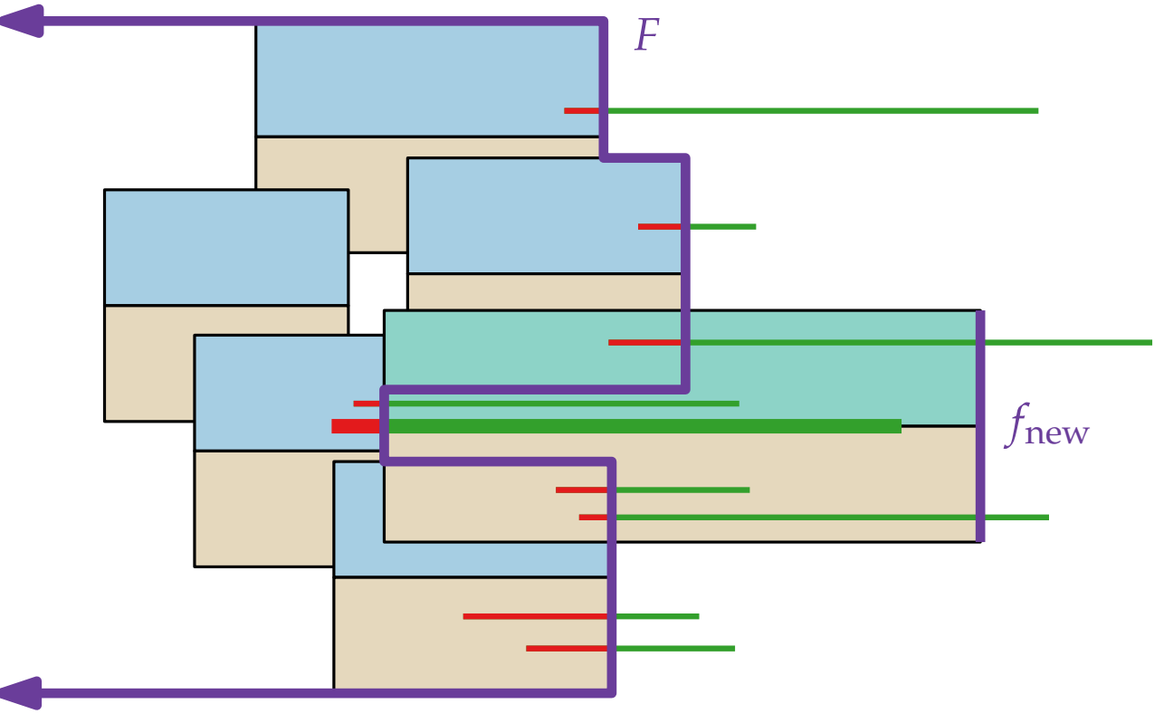
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int}

Updates



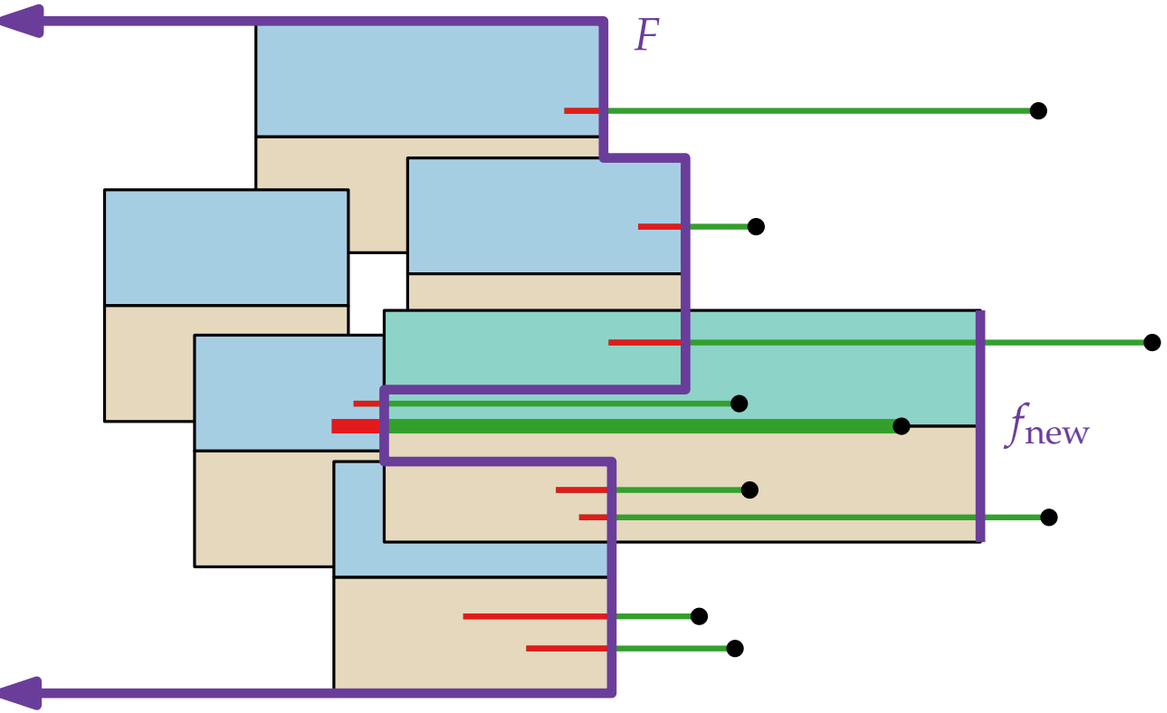
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int}

Updates



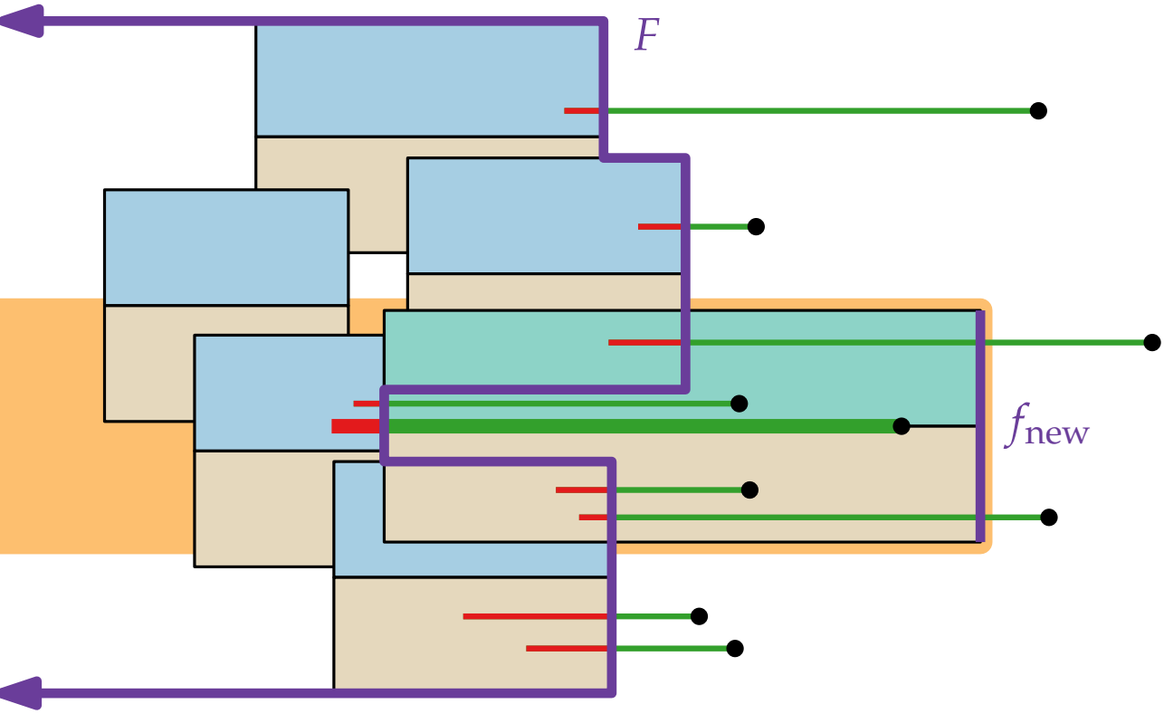
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int} ➡ nutze PST für rechte Endpunkte der Strecken

Updates



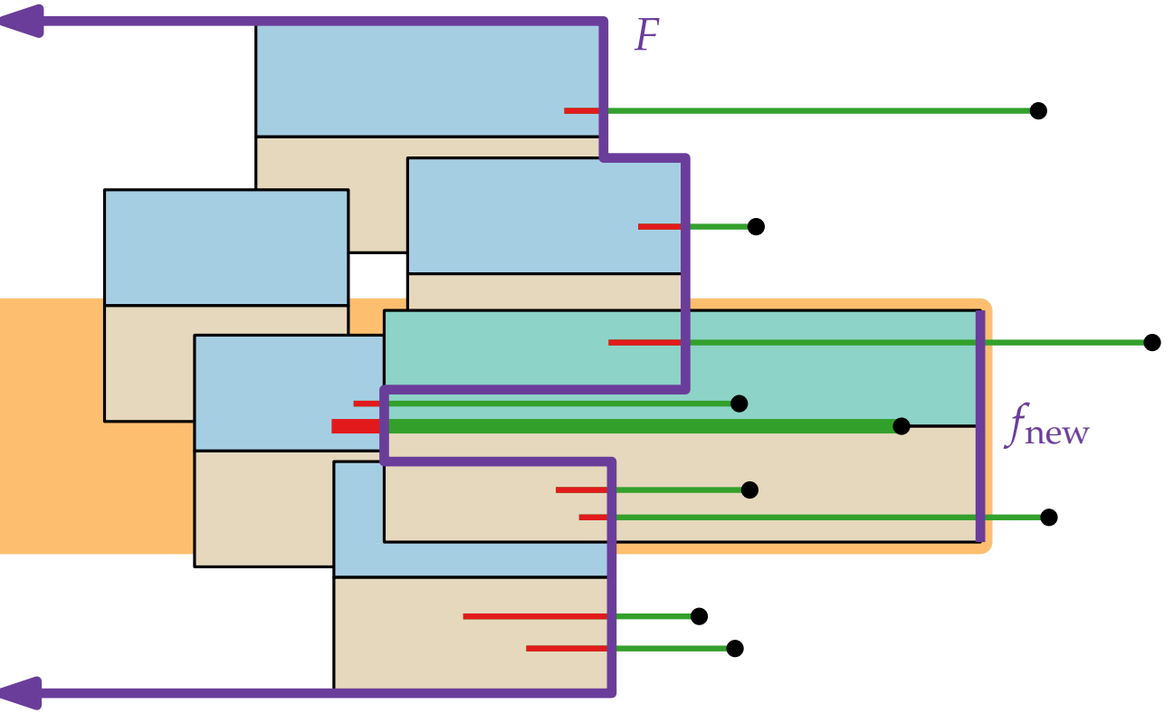
- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int} ➡ nutze PST für rechte Endpunkte der Strecken

Updates



- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int} ➡ nutze PST für rechte Endpunkte der Strecken

Updates



- Updates von H_{right} ➡ nutze PST für linke Endpunkte der Strecken
- Updates von H_{int} ➡ nutze PST für rechte Endpunkte der Strecken

Suchbäume und $\mathcal{H}_{\text{int}}$ anpassen

Der Algorithmus

von van Kreveld, Strijk und Wolff

MaxNumber 4S

AGiS

Greedy4S(P, L)

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**

- $v = \text{FindMin}(\mathcal{H}_V)$
- **while** v links von F **do**
  - $\text{DeleteMin}(\mathcal{H}_v)$
  - $v = \text{FindMin}(\mathcal{H}_V)$
- ℓ_i = linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge ℓ_i zur Beschriftung hinzu
- f_{new} = rechte Kante von $\tilde{\ell}_i$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} und update Datenstrukturen
- entferne alle Verweise auf ℓ_i aus den Datenstrukturen

Der Algorithmus

von van Kreveld, Strijk und Wolff

MaxNumber 4S

AGiS

GREEDY4S(P, L)

while $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ oder $\mathcal{H}_V$ nicht leer **do**

- $v = \text{FINDMIN}(\mathcal{H}_V)$
- **while** v links von F **do**
  - $\text{DELETEMIN}(\mathcal{H}_v)$
  - $v = \text{FINDMIN}(\mathcal{H}_V)$
- ℓ_i = linkestes Label aus $\mathcal{H}_{\text{right}}$, $\mathcal{H}_{\text{int}}$ und $\mathcal{H}_V$
- füge ℓ_i zur Beschriftung hinzu
- f_{new} = rechte Kante von $\tilde{\ell}_i$
- update F und $\mathcal{T}_V$ mit f_{new}
- suche mit der Region links von f_{new} und update Datenstrukturen
- entferne alle Verweise auf ℓ_i aus den Datenstrukturen

➡ Laufzeit $\mathcal{O}(n \log n)$, Speicher $\mathcal{O}(n)$