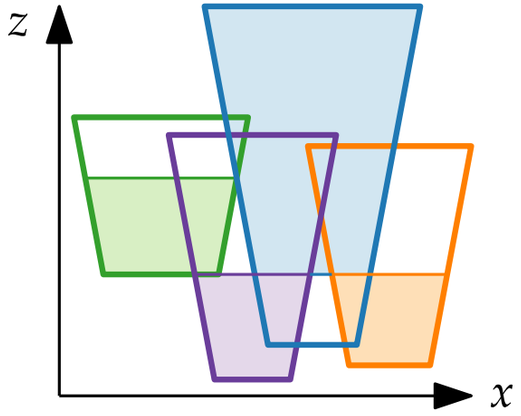
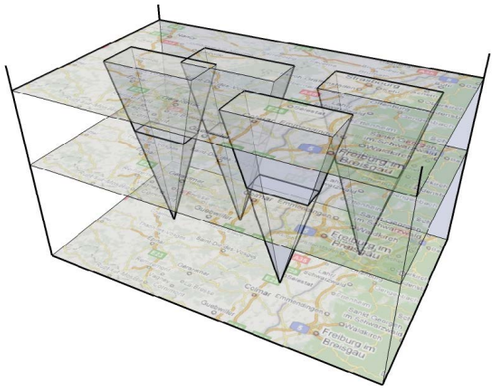
Universität Trier

Algorithmen für geographische Informationssysteme

7. Vorlesung
Dynamische Kartenbeschriftung: Zoomen



Teil I:
Dynamische Karten



Philipp Kindermann

Wintersemester 2024/25

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert
- neigt

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

➡ kontinuierliche Generalisierung

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt
- rotiert
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

- ➡ kontinuierliche Generalisierung
- ➡ kontinuierliche Kartenbeschriftung

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt
- verschiebt (einfach)
- rotiert
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

- ➡ kontinuierliche Generalisierung
- ➡ kontinuierliche Kartenbeschriftung

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt (heute)
- verschiebt (einfach)
- rotiert
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

- ➡ kontinuierliche Generalisierung
- ➡ kontinuierliche Kartenbeschriftung

Die Ära der dynamischen Karten

Die meisten Karten sind heute nicht mehr statisch und allgemein, sondern dynamisch und individuell.

Welche Eigenschaften haben dynamische Karten und wie wirkt sich das auf die Beschriftung aus?

Kartenansicht bewegt sich kontinuierlich wenn der Nutzer

- zoomt (heute)
- verschiebt (einfach)
- rotiert (nächste Woche)
- neigt

Layout und Beschriftung müssen sich an die dynamische Kartenbewegung anpassen

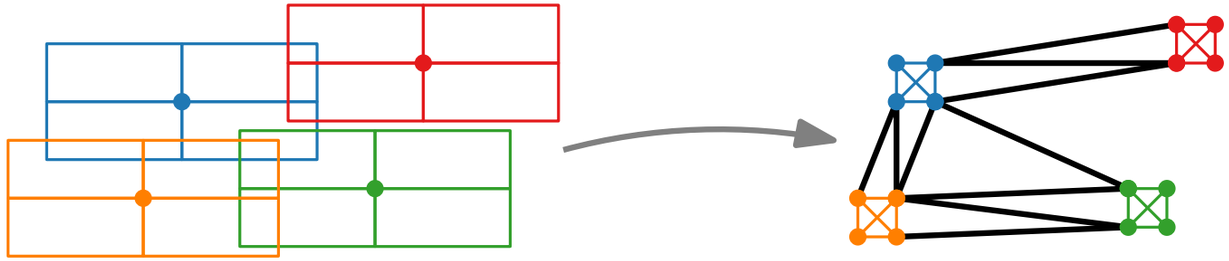
- ➡ kontinuierliche Generalisierung
- ➡ kontinuierliche Kartenbeschriftung

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.

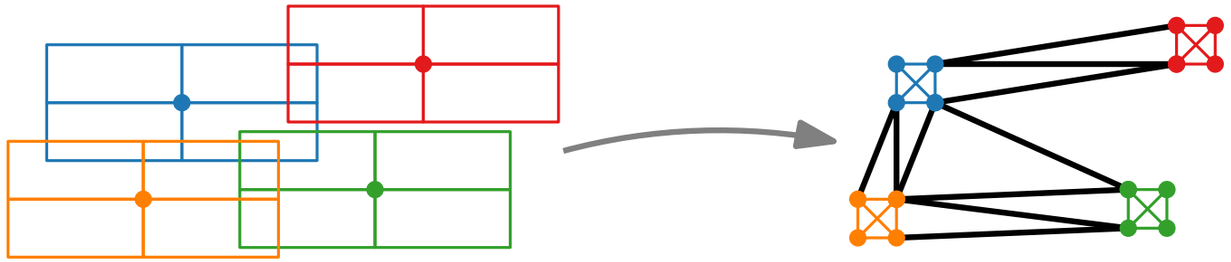
Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Lassen sich statische Verfahren nutzen?

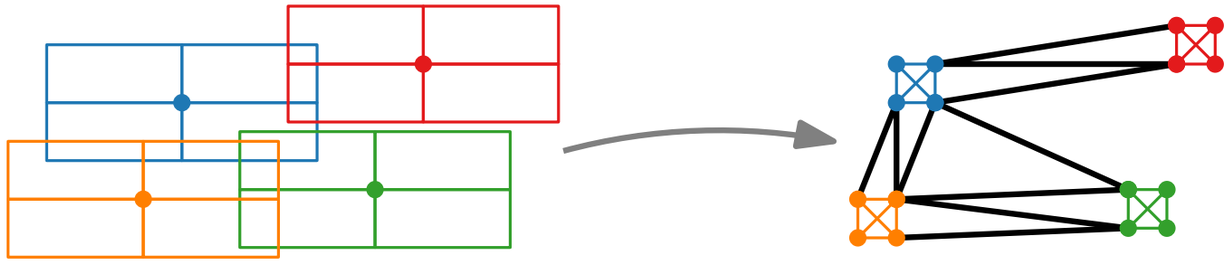
Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.

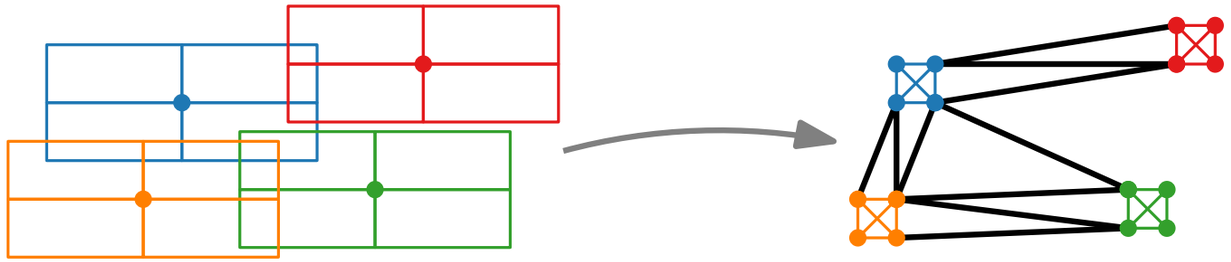


Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer ’03]

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



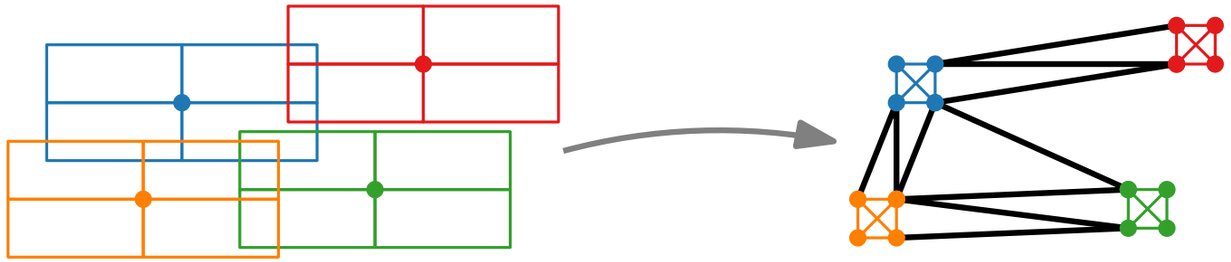
Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer '03]

- konstruiere reaktiven Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

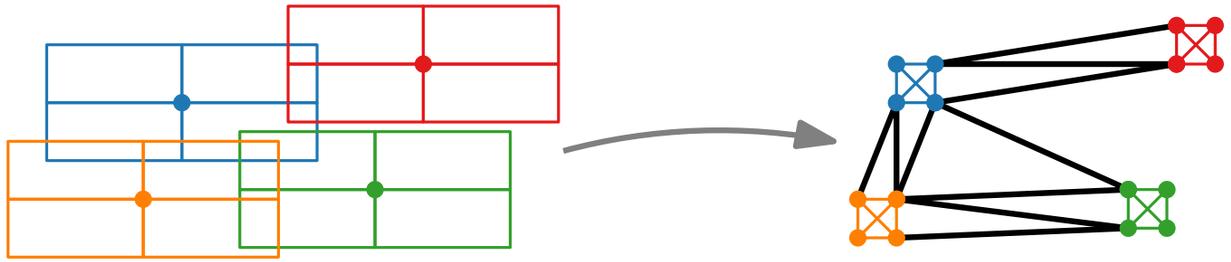
Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer '03]

- konstruiere reaktiven Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

Ansatz 2: Schnelle on-the-fly Berechnung

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer '03]

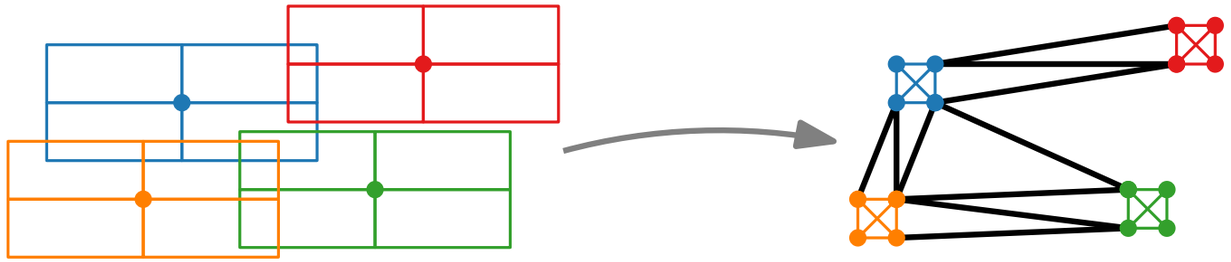
- konstruiere reaktiven Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

Ansatz 2: Schnelle on-the-fly Berechnung

- berechne lokalen Konfliktgraph, schätze Positionskosten, suche greedy günstigste Position [Mote '07]

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer '03]

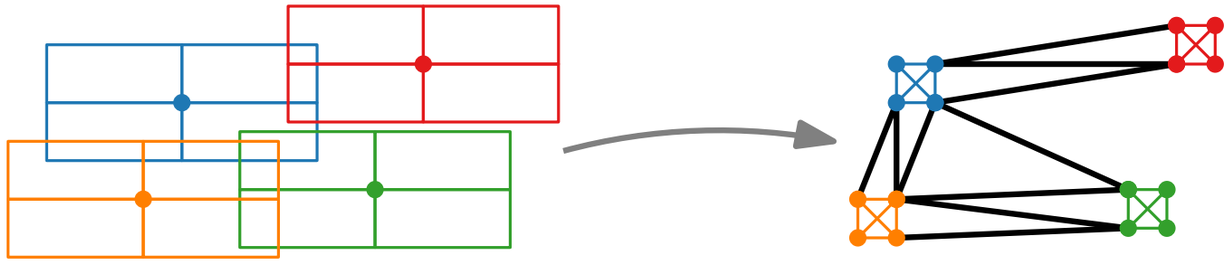
- konstruiere reaktiven Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

Ansatz 2: Schnelle on-the-fly Berechnung

- berechne lokalen Konfliktgraph, schätze Positionskosten, suche greedy günstigste Position [Mote '07]
- partikelbasierte Konflikterkennung, sequentielle Platzierung nach absteigender Positionspräferenz [Luboschik, Schumann & Cords '08]

Lassen sich statische Verfahren nutzen?

Die meisten heuristischen **statischen** Ansätze nutzen Konfliktgraph als Modell für Überlappungen und suchen große unabhängige Labelmenge.



Übertragung auf **dynamische** Karten:
schnelle Algorithmen um jeden Frame einer Animation in Echtzeit zu beschriften.

Ansatz 1: Vorberechnung & Abfragen [Petzold, Gröger & Plümer '03]

- konstruiere reaktiven Konfliktgraph, Abfrage während Interaktion, verwende greedy Verfahren für statische Labelauswahl

Ansatz 2: Schnelle on-the-fly Berechnung

Ist das Problem damit gelöst?

- berechne lokalen Konfliktgraph, schätze Positionskosten, suche greedy günstigste Position [Mote '07]
- partikelbasierte Konflikterkennung, sequentielle Platzierung nach absteigender Positionspräferenz [Luboschik, Schumann & Cords '08]

Neue Verfahren

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

Neue Verfahren

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet

Neue Verfahren

Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen

Neue Verfahren

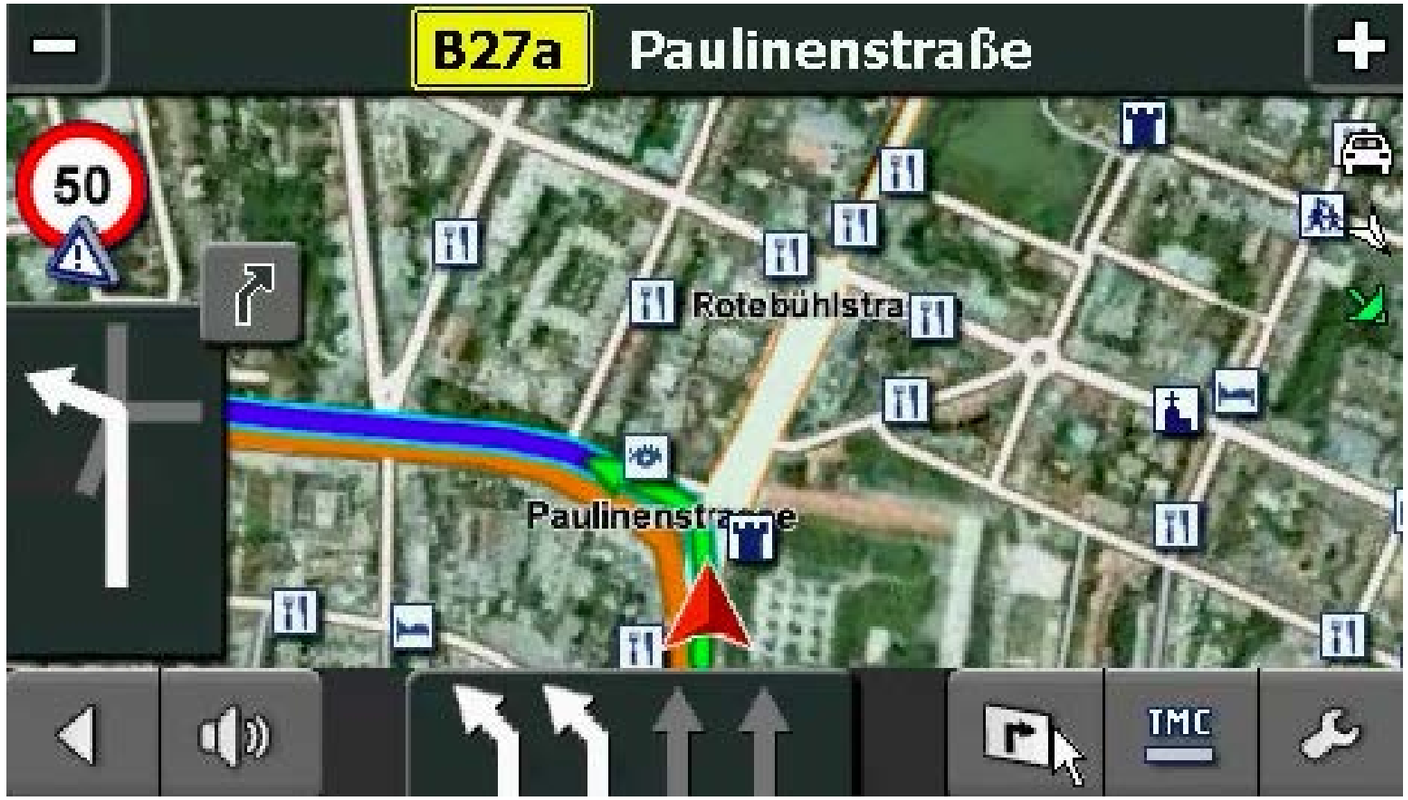
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

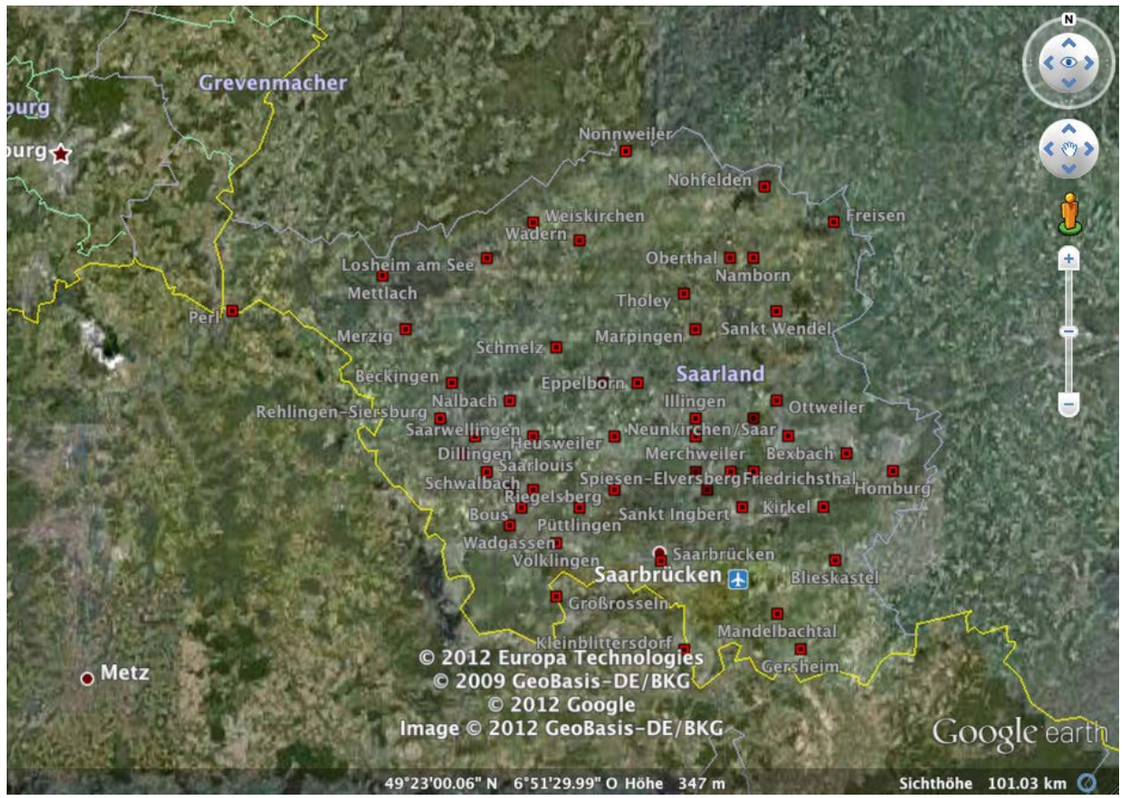
- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet

Neue Verfahren

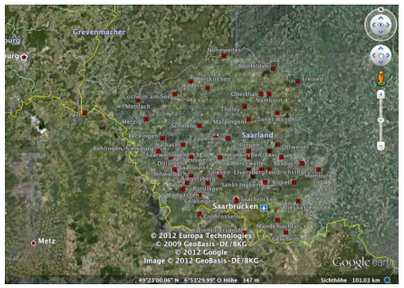
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet





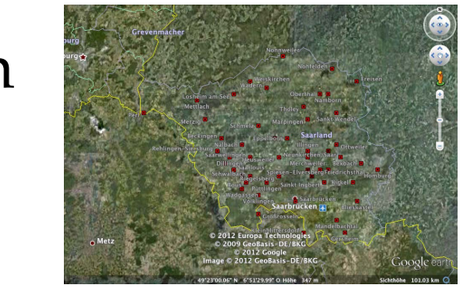
Neue Verfahren



Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

Neue Verfahren



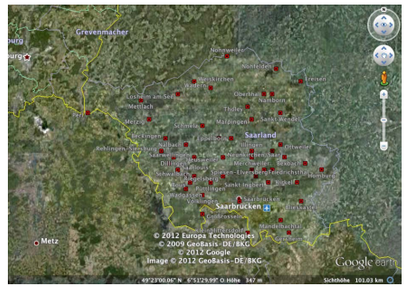
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

Konsistenzkriterien:

[Been, Daiches & Yap ’06]

Neue Verfahren



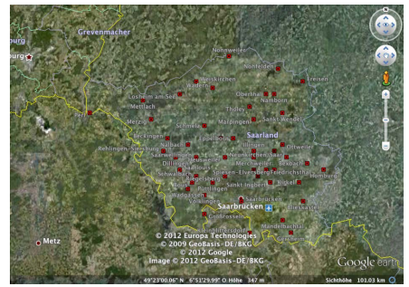
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

Konsistenzkriterien: [Been, Daiches & Yap '06]

- **Monotonität:** Label sollen beim Vergrößern nicht verschwinden und beim Verkleinern nicht auftauchen

Neue Verfahren



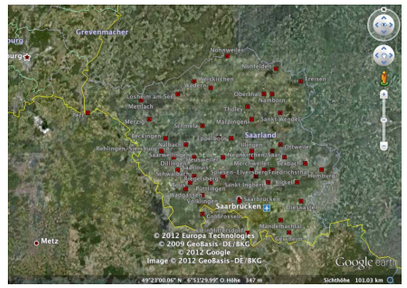
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

Konsistenzkriterien: [Been, Daiches & Yap '06]

- **Monotonität:** Label sollen beim Vergrößern nicht verschwinden und beim Verkleinern nicht auftauchen
- **Kontinuität:** sichtbare Label sollen Position und Größe kontinuierlich ändern

Neue Verfahren



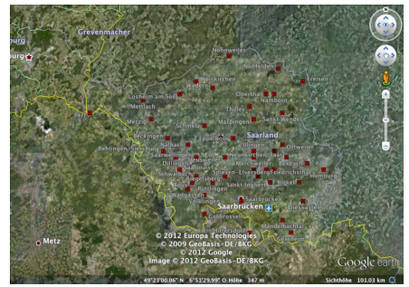
Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

Konsistenzkriterien: [Been, Daiches & Yap '06]

- **Monotonität:** Label sollen beim Vergrößern nicht verschwinden und beim Verkleinern nicht auftauchen
- **Kontinuität:** sichtbare Label sollen Position und Größe kontinuierlich ändern
- **Konsistenz:** Platzierung und Auswahl der Label soll eine Funktion des Kartenzustands sein und nicht von der Historie abhängen

Neue Verfahren



Die bisherigen Ansätze sind zwar schnell genug, aber die resultierenden Kartenanimationen sind oft unbefriedigend.

- jeder Frame wird unabhängig von Nachbarframes beschriftet
- Label neigen zu Flackern und Sprüngen
- **zeitliche Kohärenz** oder **Konsistenz** wird nicht betrachtet
- aktuelle reale Beispiele zeigen noch immer negative Effekte!

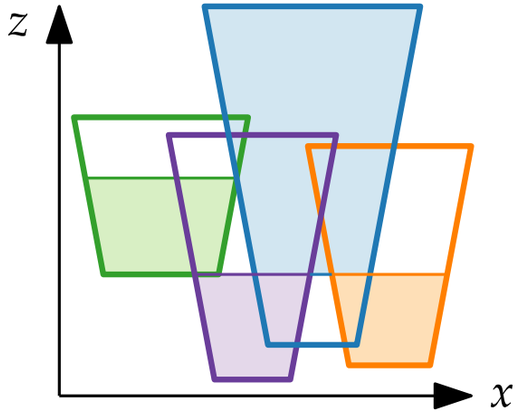
Konsistenzkriterien: [Been, Daiches & Yap '06]

- **Monotonität:** Label sollen beim Vergrößern nicht verschwinden und beim Verkleinern nicht auftauchen
- **Kontinuität:** sichtbare Label sollen Position und Größe kontinuierlich ändern
- **Konsistenz:** Platzierung und Auswahl der Label soll eine Funktion des Kartenzustands sein und nicht von der Historie abhängen
- **feste Labelgröße** (auf dem Bildschirm)

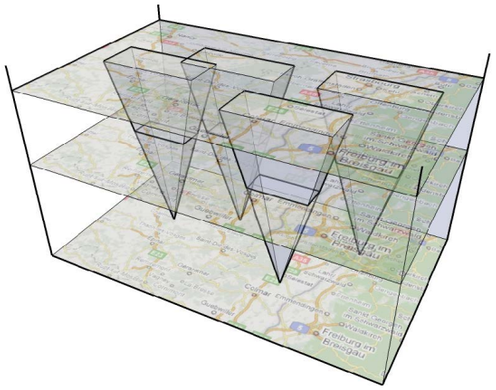
Universität Trier

Algorithmen für geographische Informationssysteme

7. Vorlesung
Dynamische Kartenbeschriftung: Zoomen



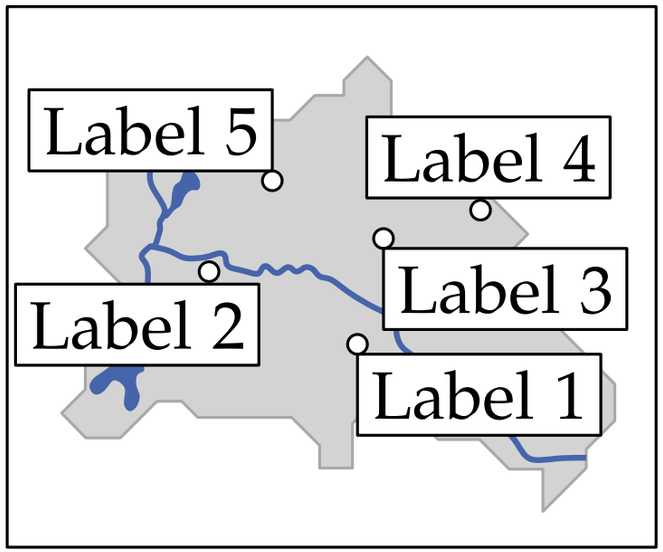
Teil II:
Modell für Beschriften mit Zoom



Philipp Kindermann

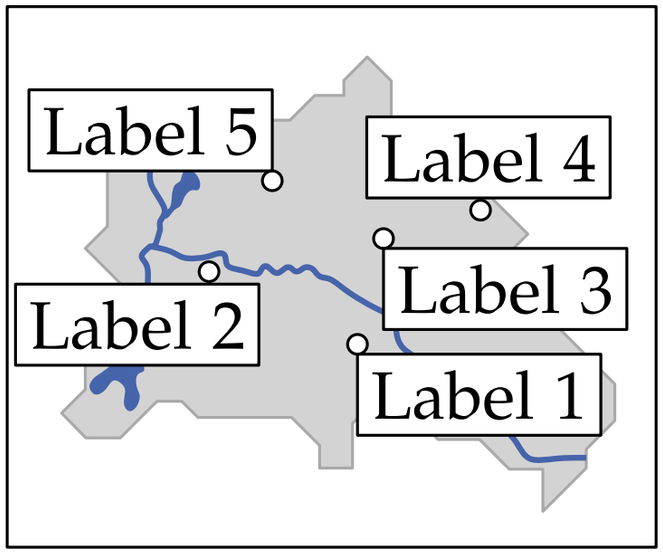
Wintersemester 2024/25

Label in Karten mit Zoomfunktion

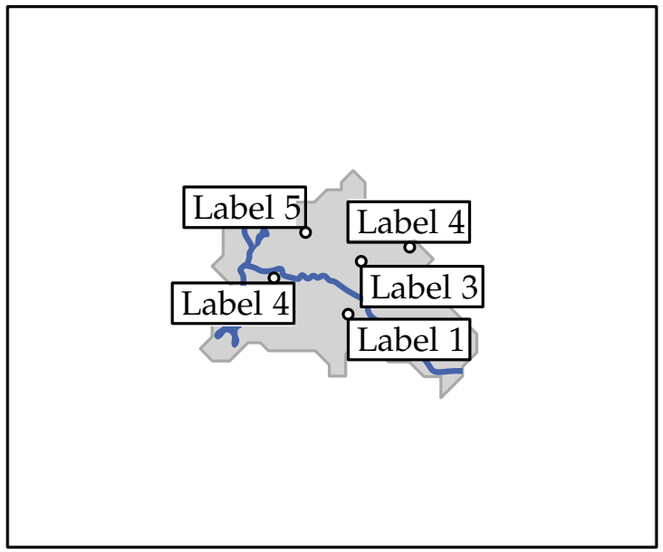


großer Maßstab

Label in Karten mit Zoomfunktion

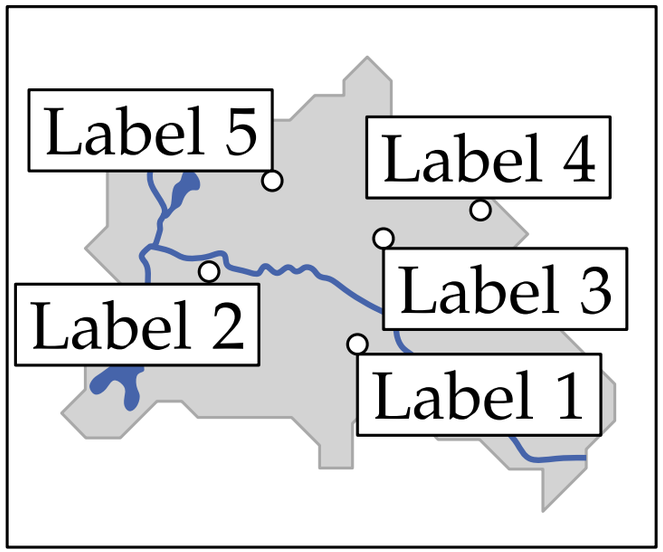


großer Maßstab

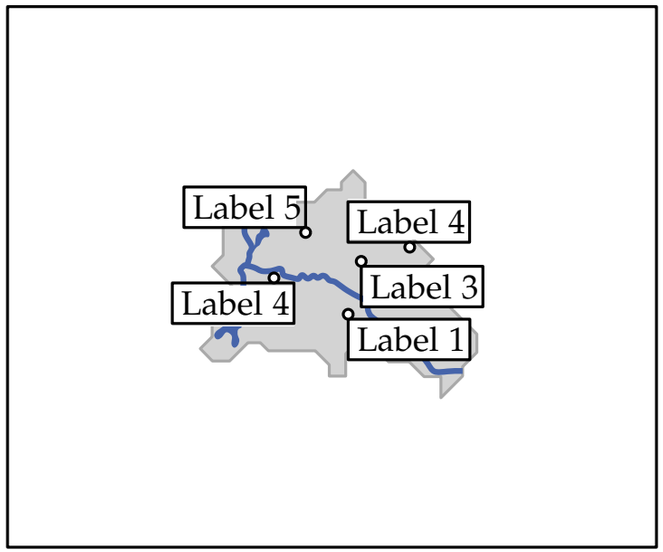


kleiner Maßstab

Label in Karten mit Zoomfunktion



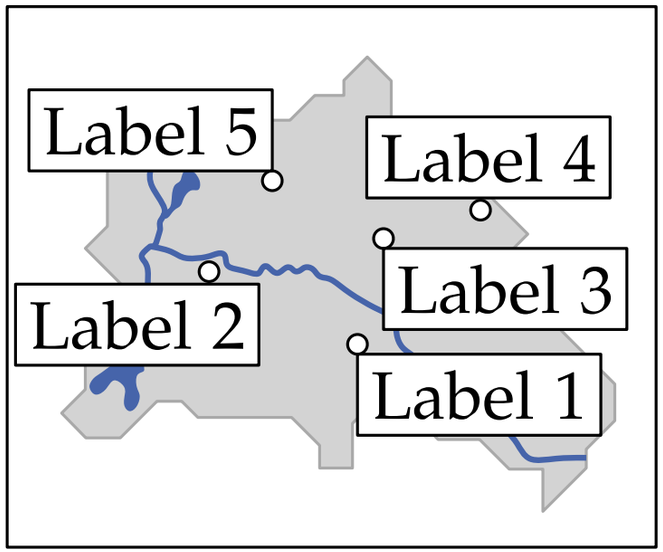
großer Maßstab



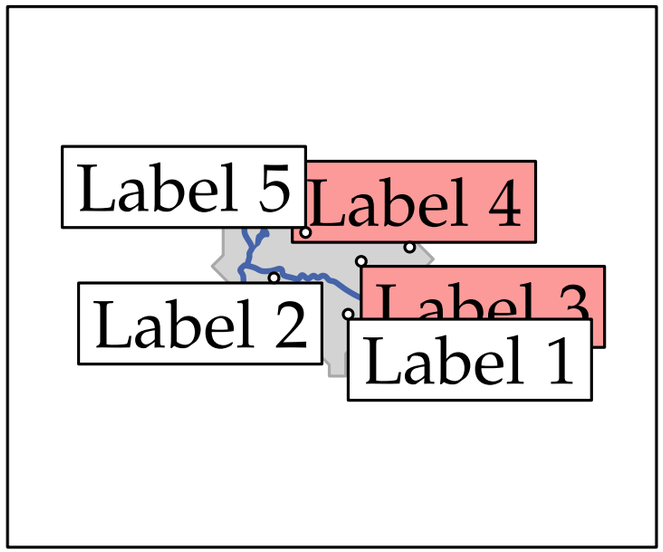
kleiner Maßstab

- feste Labelgröße

Label in Karten mit Zoomfunktion



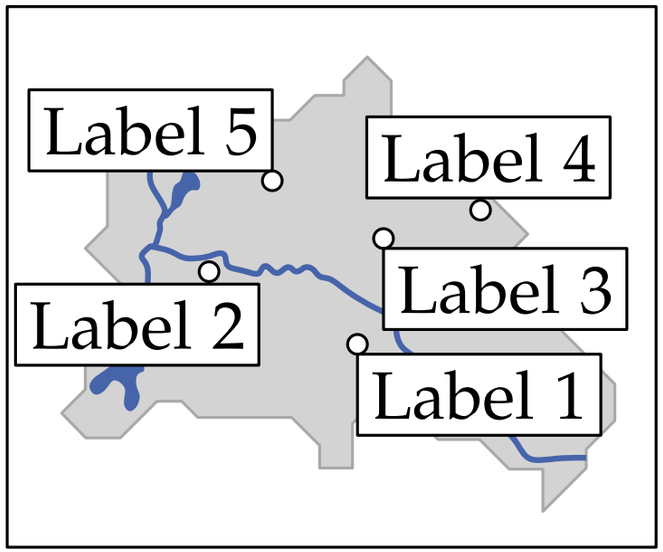
großer Maßstab



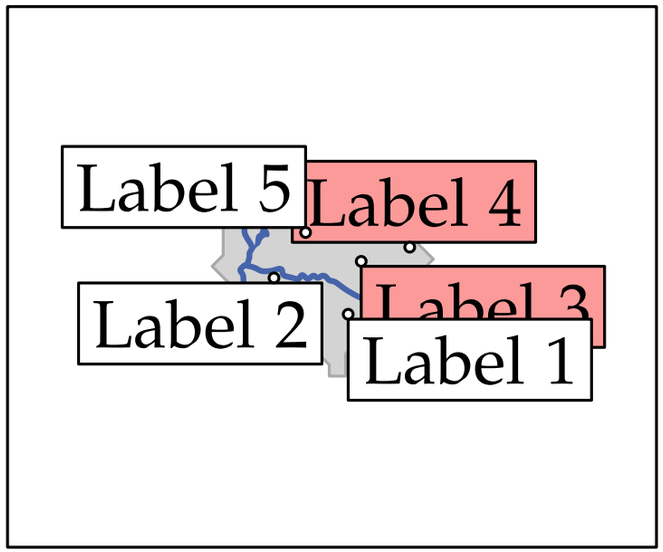
kleiner Maßstab

- feste Labelgröße

Label in Karten mit Zoomfunktion



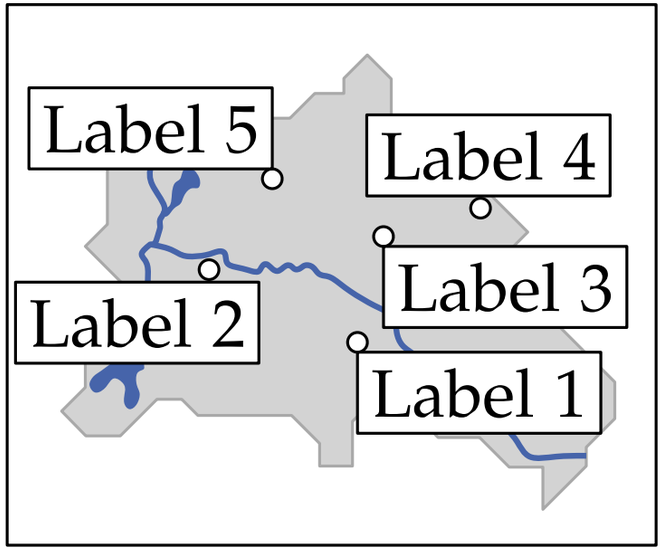
großer Maßstab



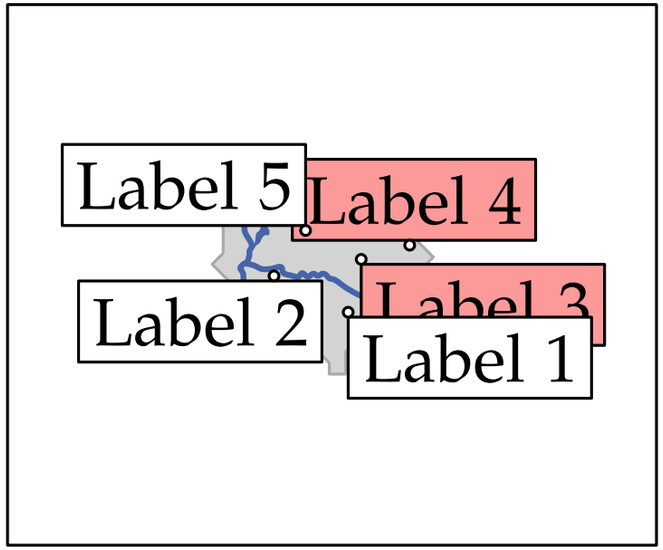
kleiner Maßstab

- feste Labelgröße ➡ neue Konflikte

Label in Karten mit Zoomfunktion



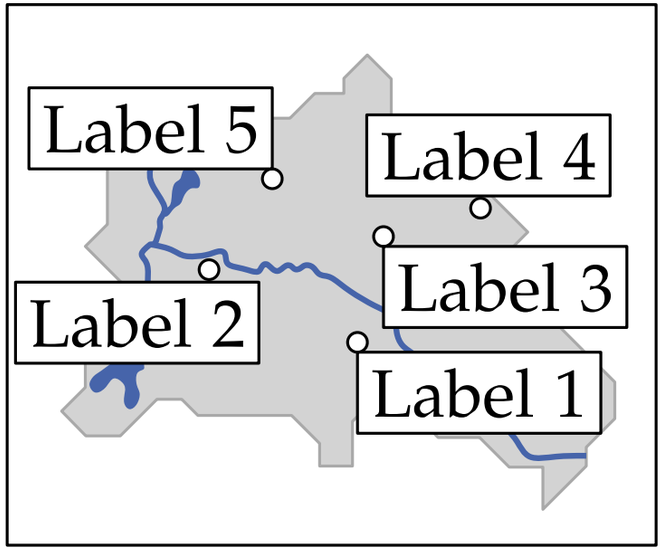
großer Maßstab



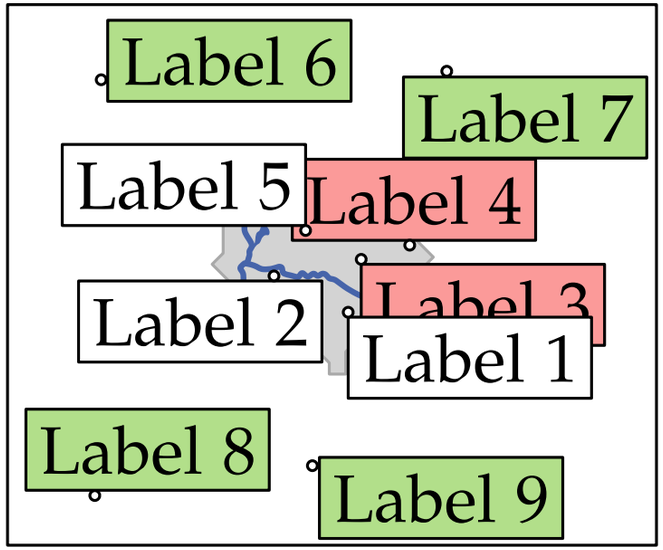
kleiner Maßstab

- feste Labelgröße ➡ neue Konflikte
- neue Punkte kommen in Kartenausschnitt hinzu

Label in Karten mit Zoomfunktion



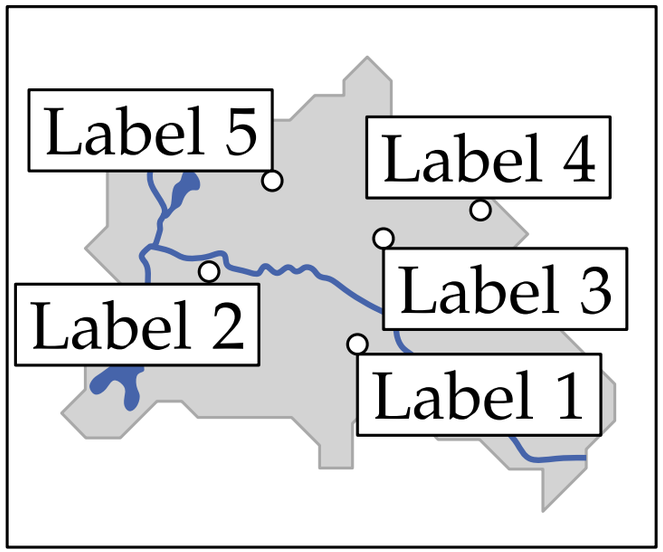
großer Maßstab



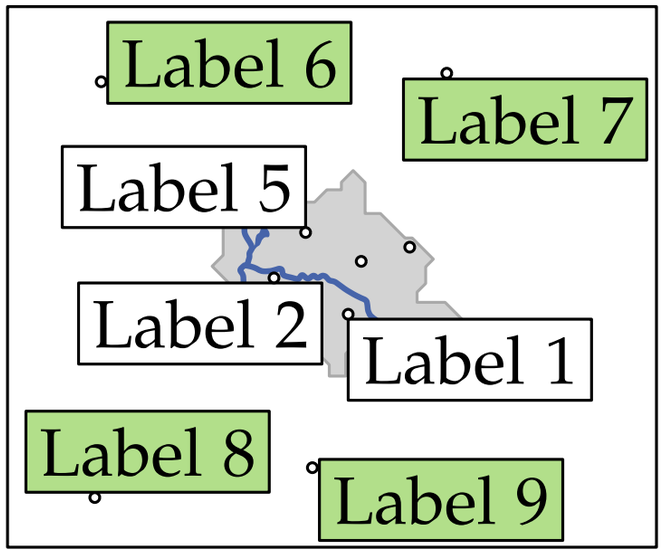
kleiner Maßstab

- feste Labelgröße ➡ neue Konflikte
- neue Punkte kommen in Kartenausschnitt hinzu

Label in Karten mit Zoomfunktion



großer Maßstab

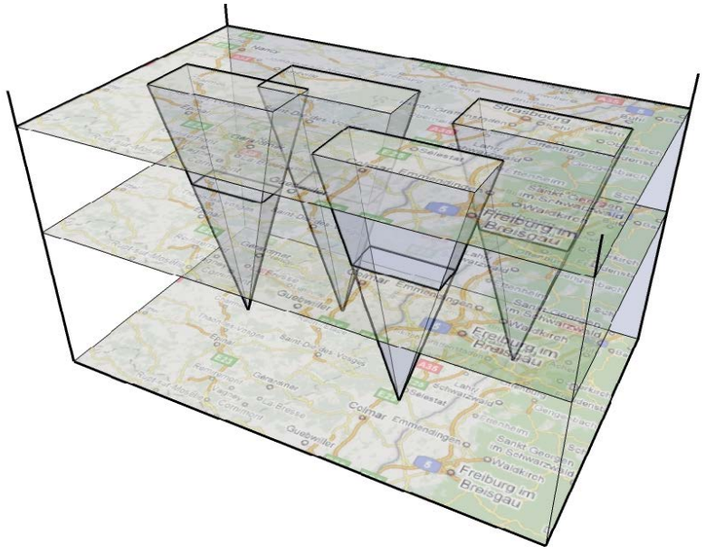


kleiner Maßstab

- feste Labelgröße ➡ neue Konflikte
- neue Punkte kommen in Kartenausschnitt hinzu

3D Modell für dynamisches Beschriften

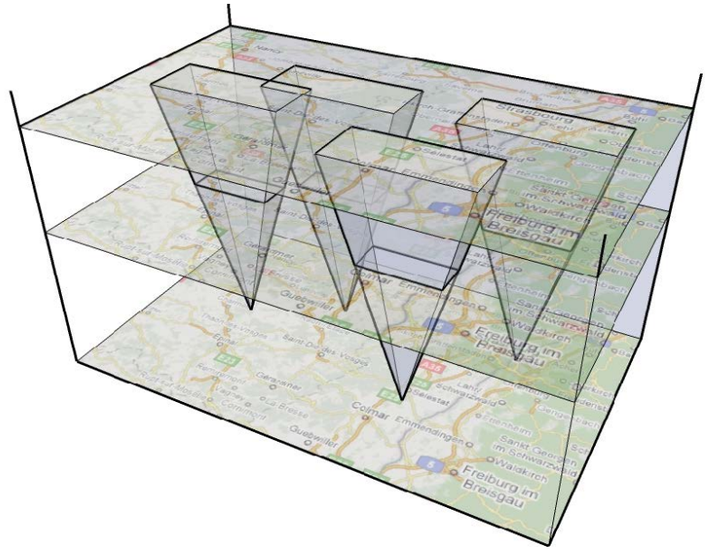
[Been, Daiches & Yap ’06]



3D Modell für dynamisches Beschriften

[Been, Daiches & Yap ’06]

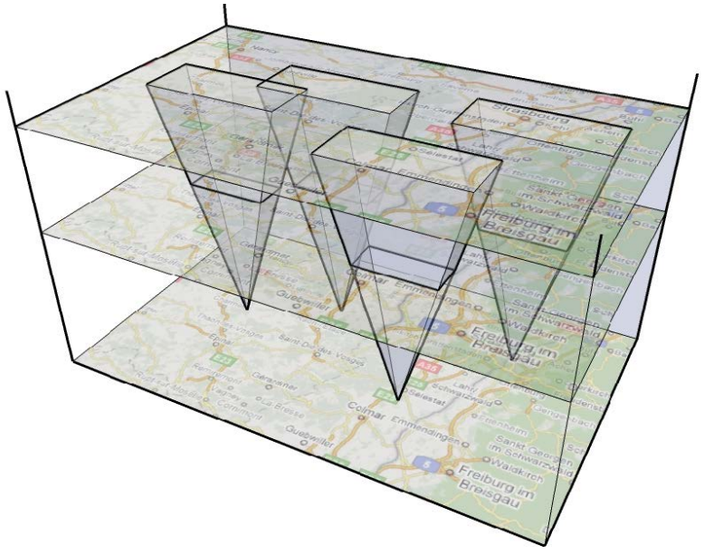
- (inverser) Maßstab auf z-Achse



3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]

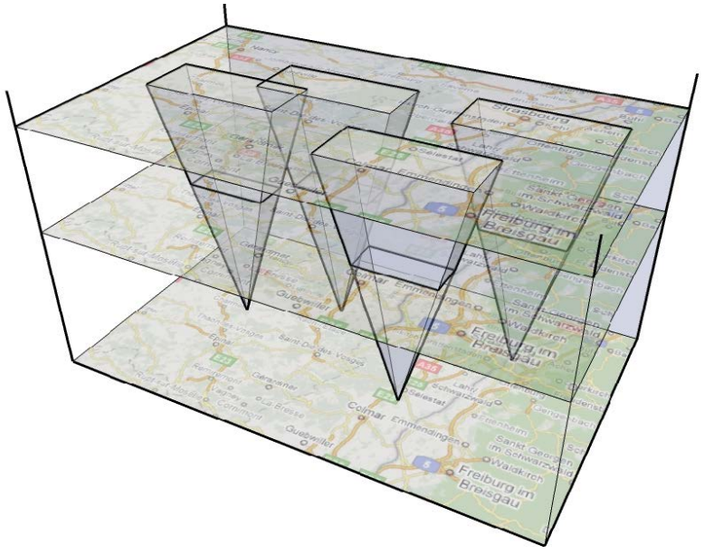
- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$



3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]

- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

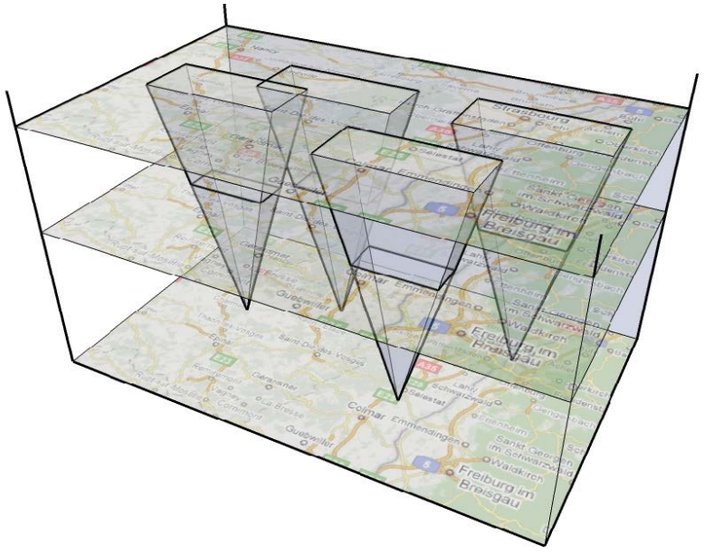


3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]

- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

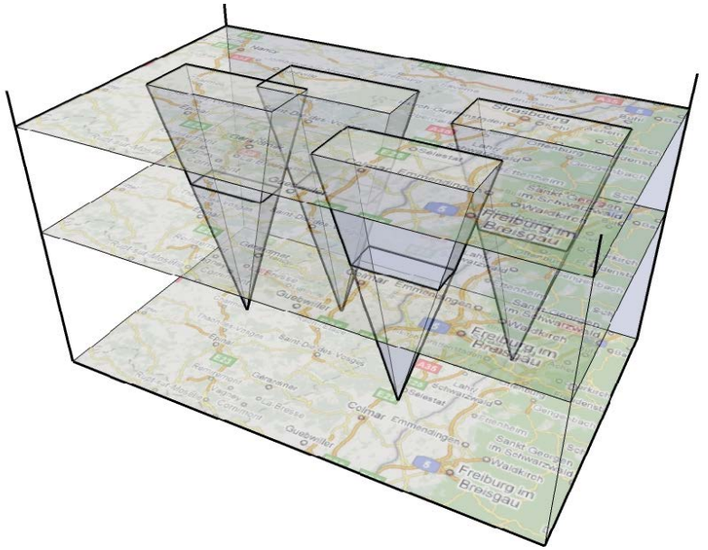


3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]

- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**



Betrachte erstmal 1D-Variante.

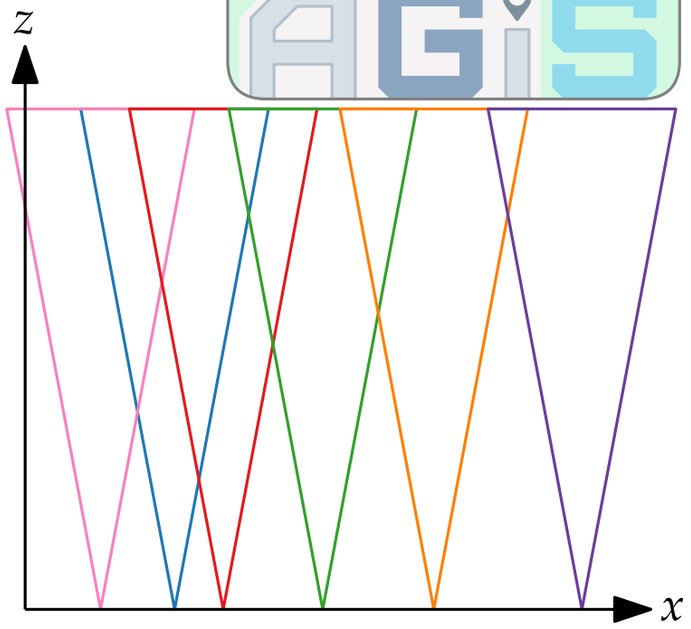
3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]

- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

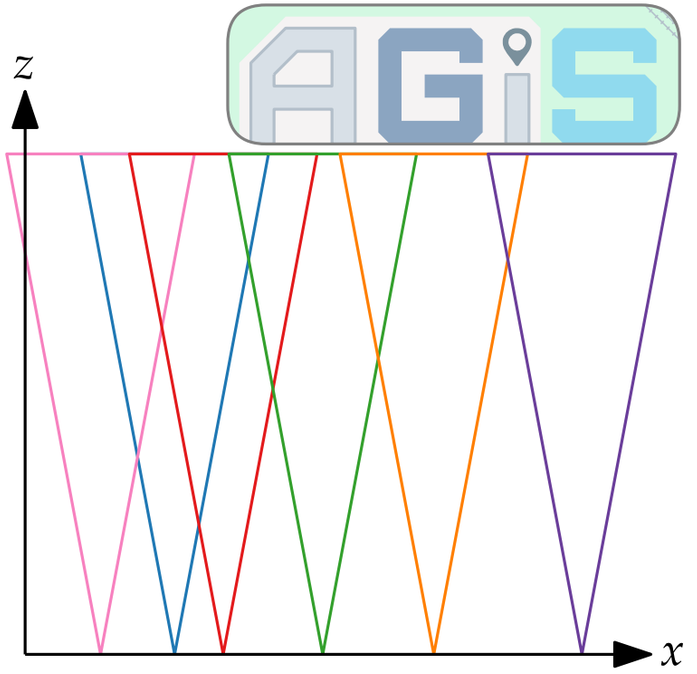
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.



3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

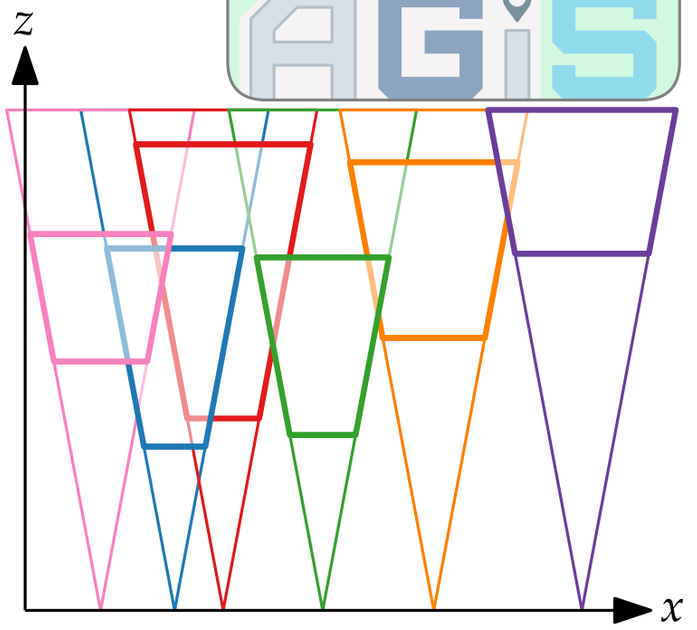
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

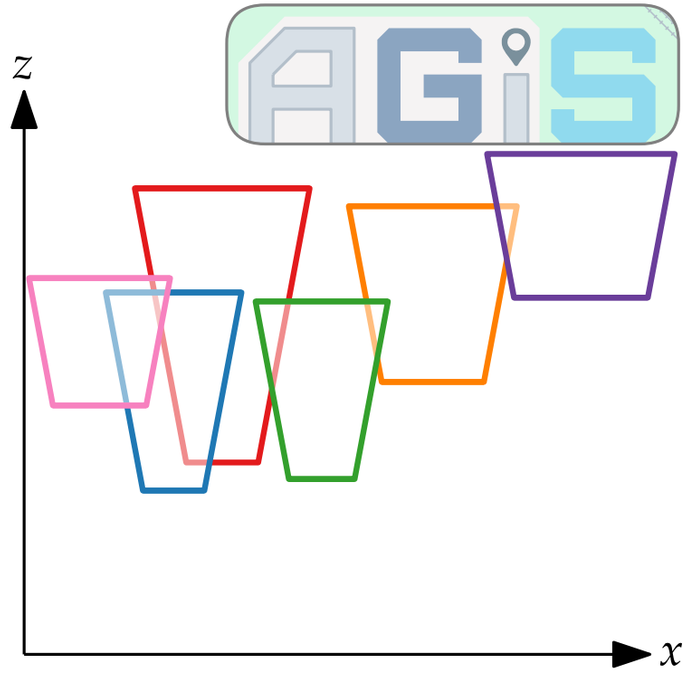
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

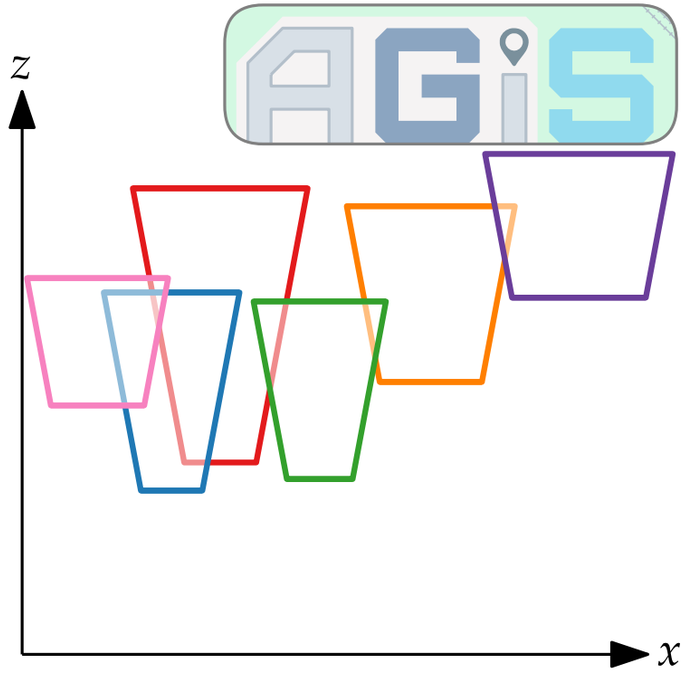
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

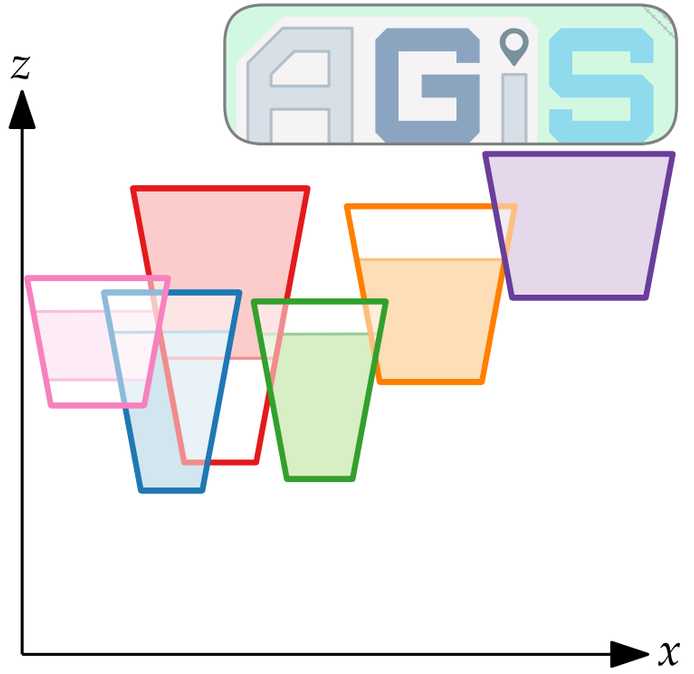
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

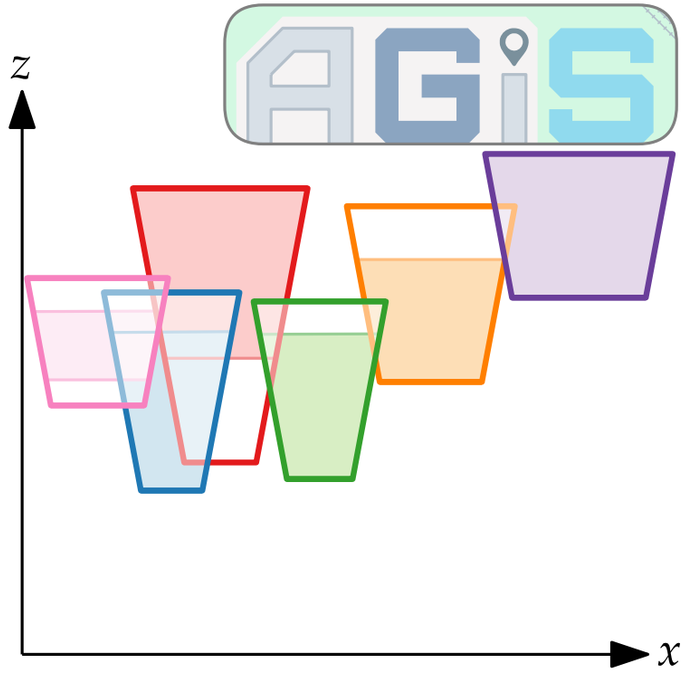
  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

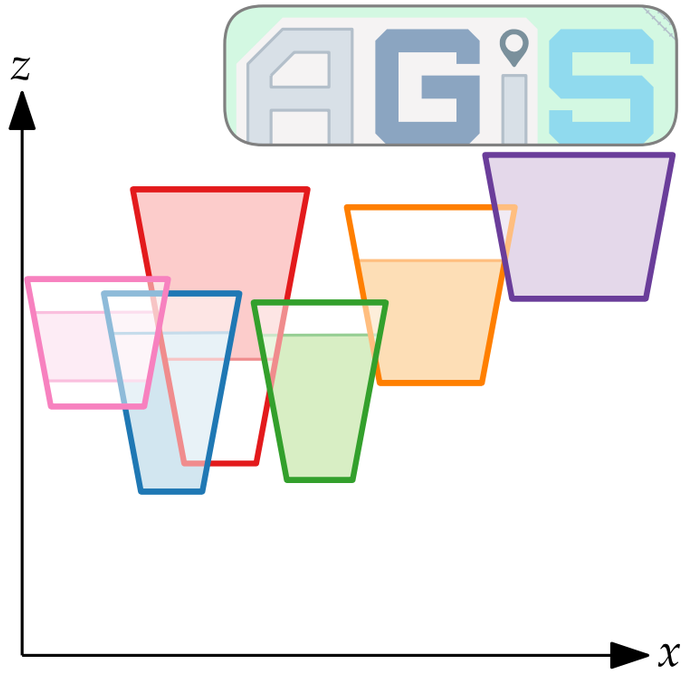
Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

  ➡ **kein Flackern**

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

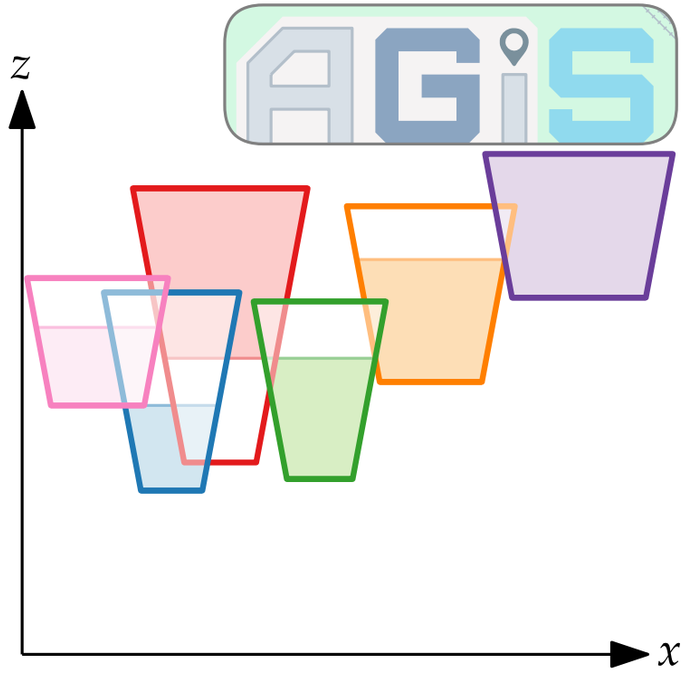
Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

  ➡ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (Labelkörper) induzieren

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

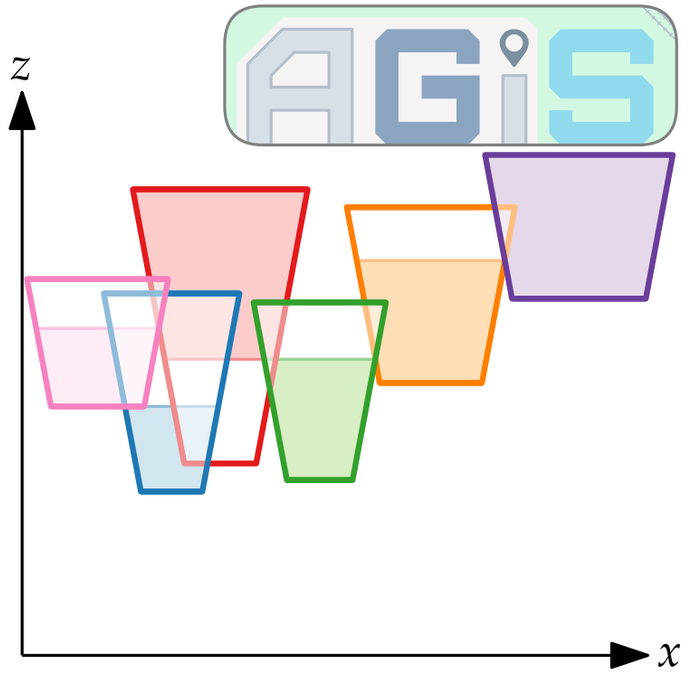
Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

  ➡ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (Labelkörper) induzieren

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

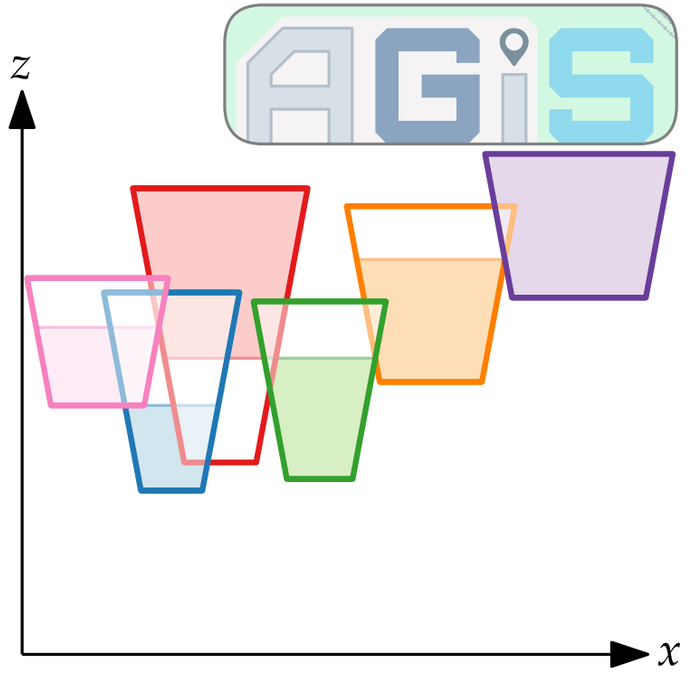
- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

  ➡ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (Labelkörper) induzieren

  ➡ **keine Überlappungen**

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

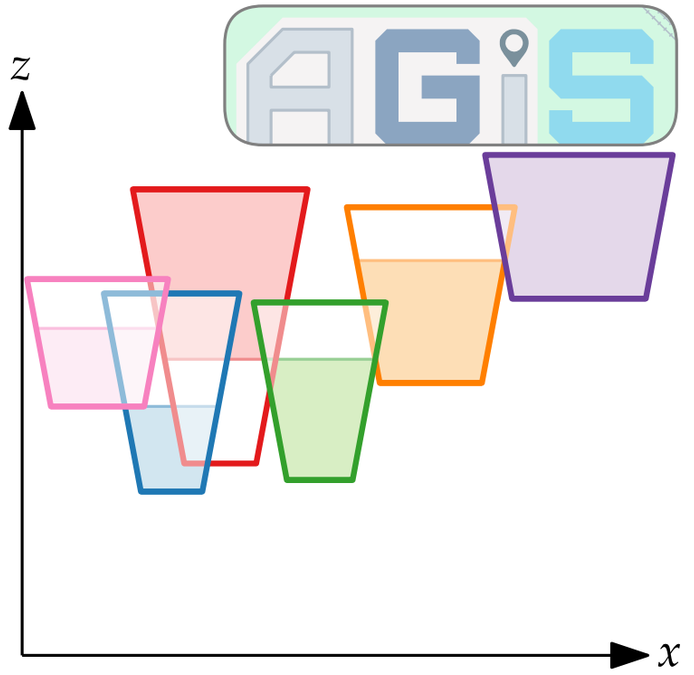
  ➡ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (Labelkörper) induzieren

  ➡ **keine Überlappungen**

Ziel:

3D Modell für dynamisches Beschriften

[Been, Daiches & Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontaler Schnitt bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  ➡ **kein Springen**

Betrachte erstmal 1D-Variante.

- Jedes Label L hat ein verfügbares Intervall S_L (z.B. nach Wichtigkeit)
- berechne ein **aktives Intervall** $A_L \subseteq S_L$

  ➡ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (Labelkörper) induzieren

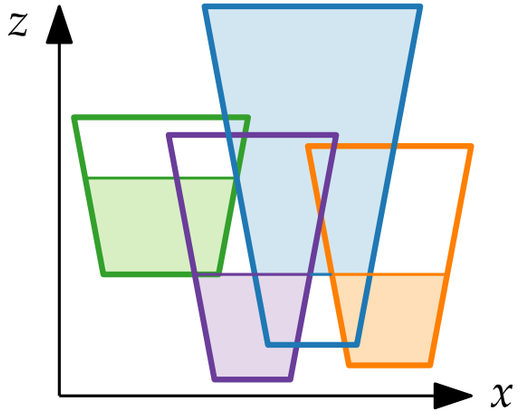
  ➡ **keine Überlappungen**

Ziel: maximiere **aktive Gesamthöhe** $\sum_L |A_L|$

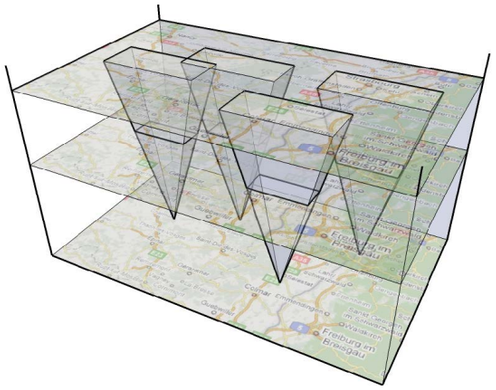
Universität Trier

Algorithmen für geographische Informationssysteme

7. Vorlesung
Dynamische Kartenbeschriftung: Zoomen



Teil III:
Greedy Top-Down Algorithmus



Philipp Kindermann

Wintersemester 2024/25

Greedy Top-Down Sweep Algorithmus

Gegeben:

Gegeben:

Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden

Gegeben:

Greedy Top-Down Sweep Algorithmus

Gegeben:

- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben:

Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)

Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

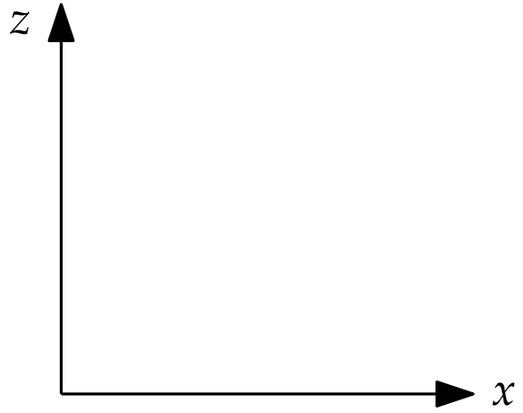
GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$

Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$

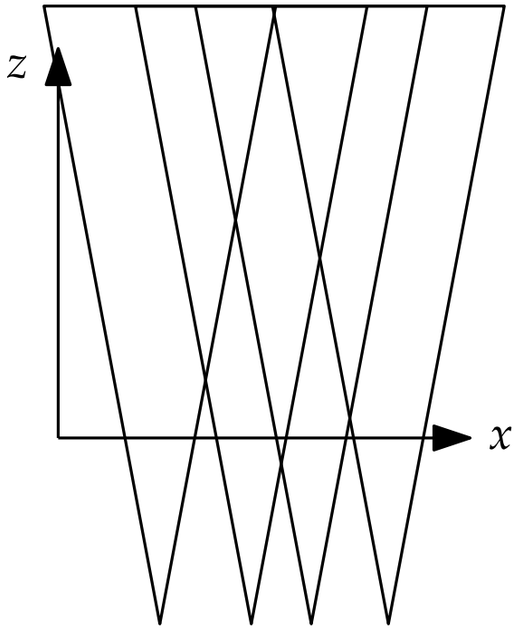


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$

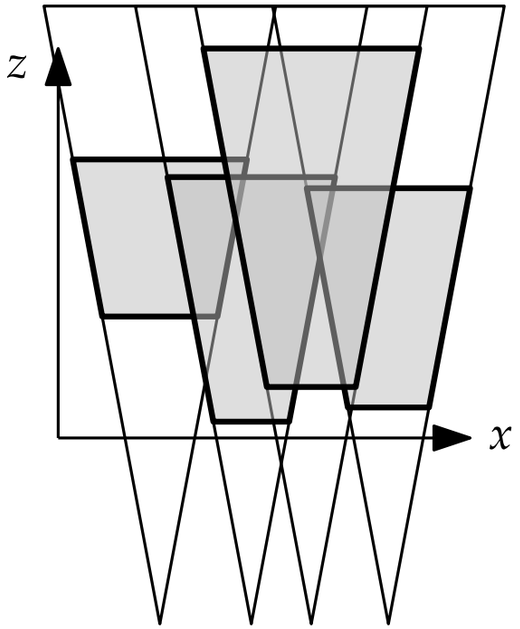


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$

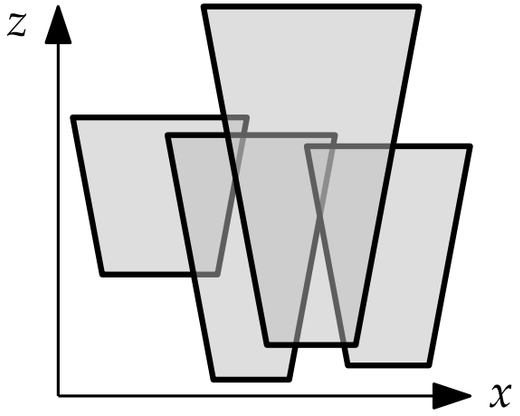


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$

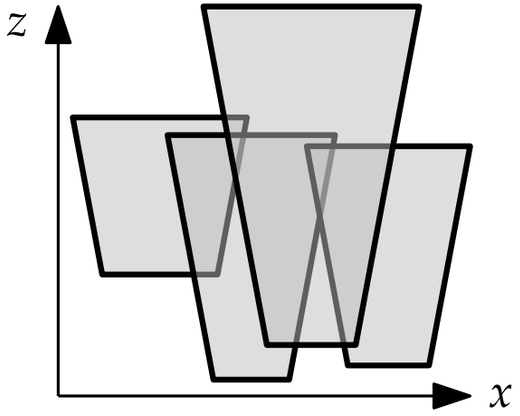


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$
  $Q =$ PriorityQueue($\mathcal{L}$)

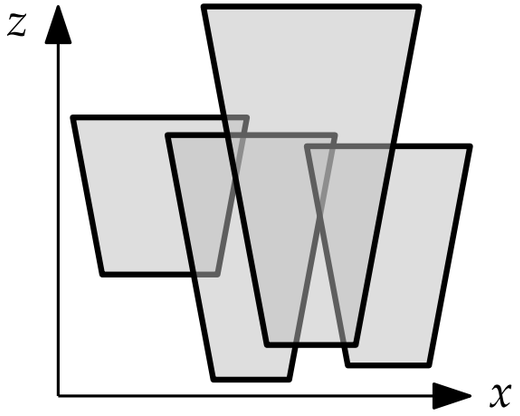


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$
  Q = PriorityQueue($\mathcal{L}$) absteigend sortiert nach (A_L, S_L)

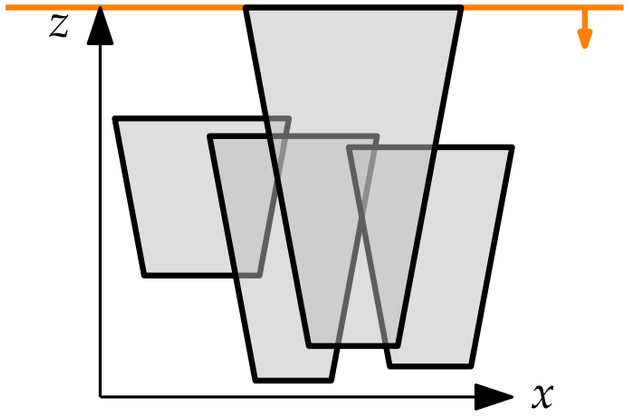


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

GreedyTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $(a_L, A_L) = (s_L, S_L)$
  $Q =$ PriorityQueue($\mathcal{L}$) absteigend sortiert nach (A_L, S_L)

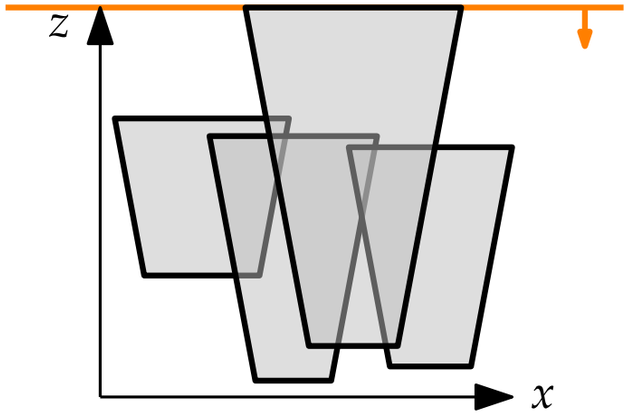


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
```

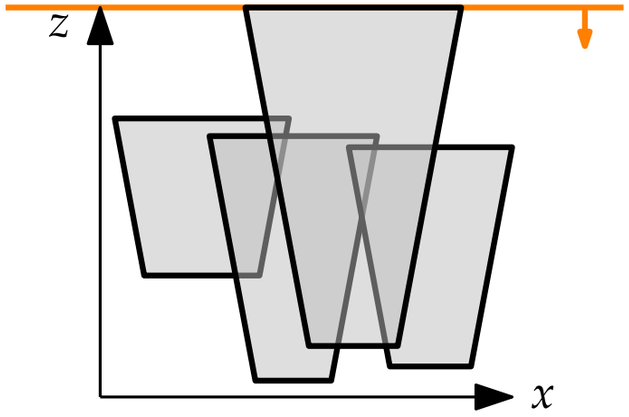


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)   absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
```

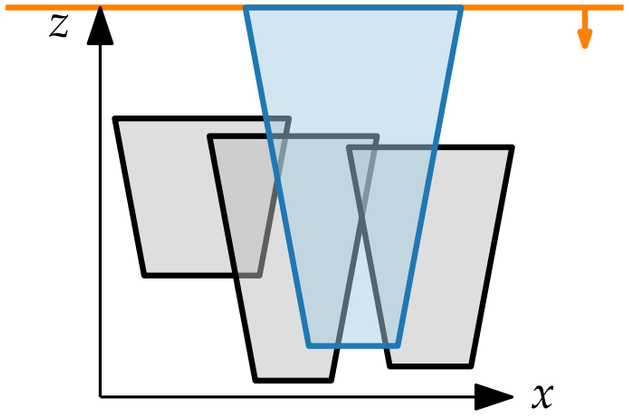


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
```

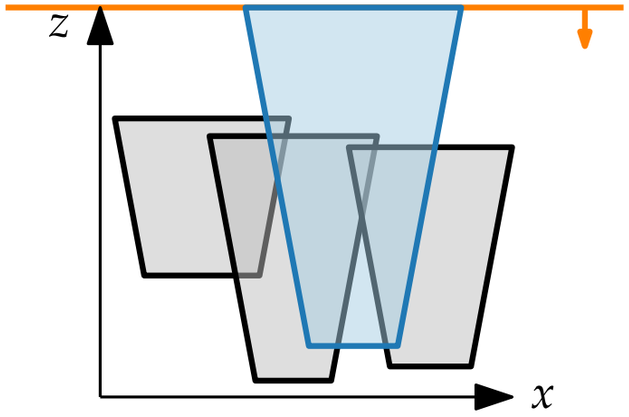


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
```

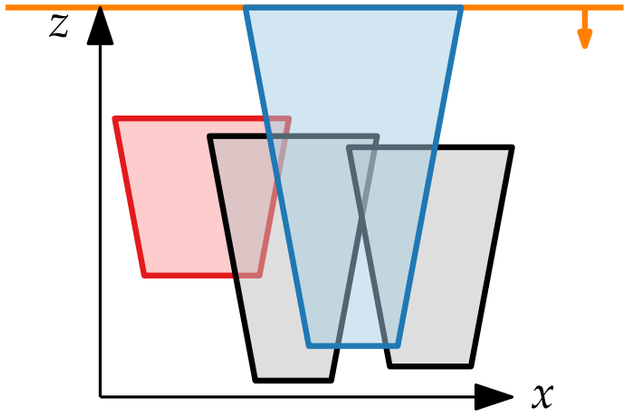


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
```

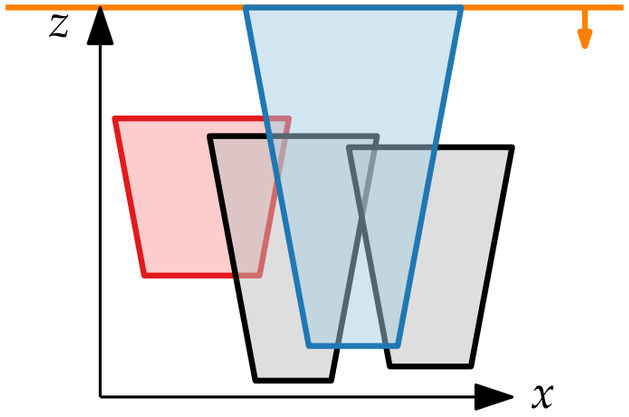


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
```

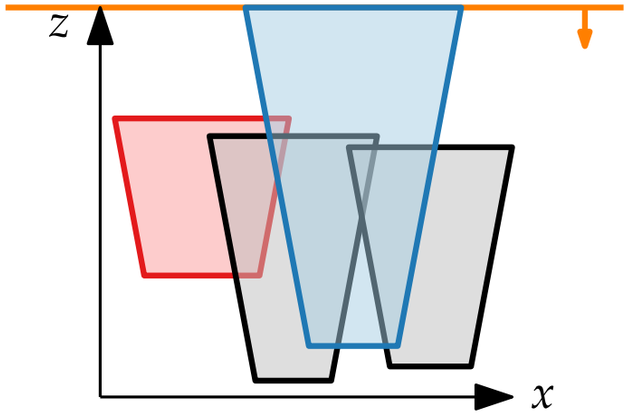


Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

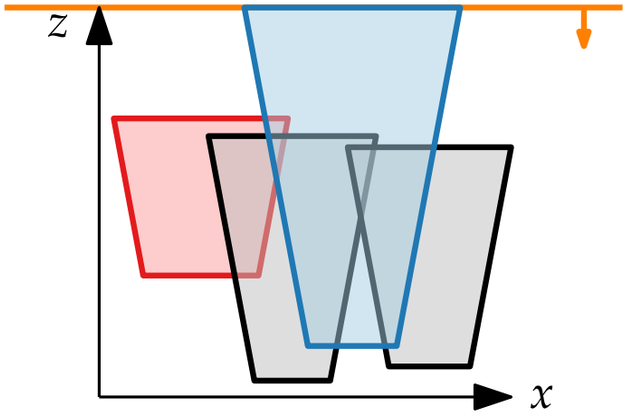
```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
```



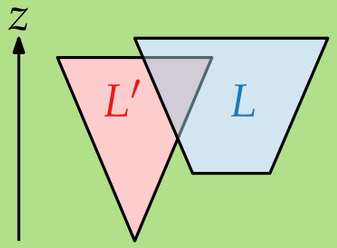
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



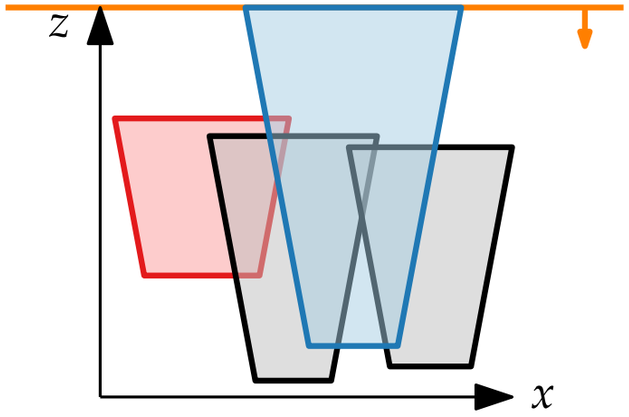

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
```



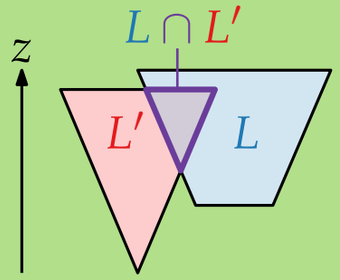
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



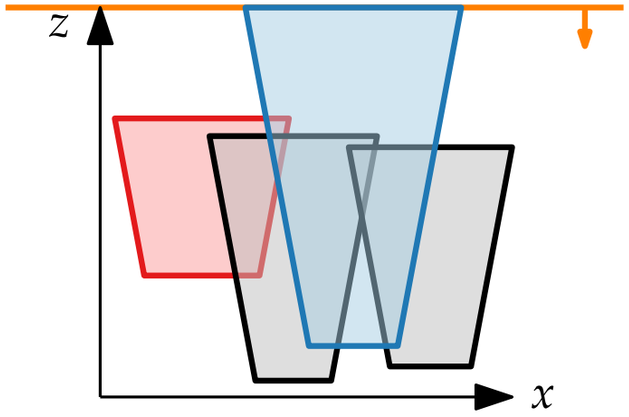

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
```



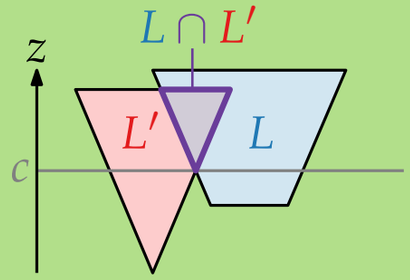
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



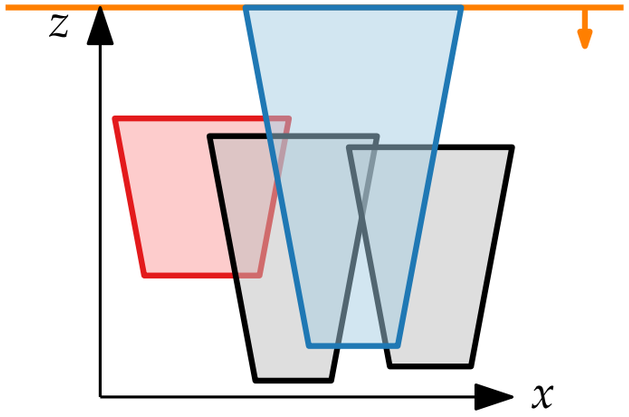

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
```



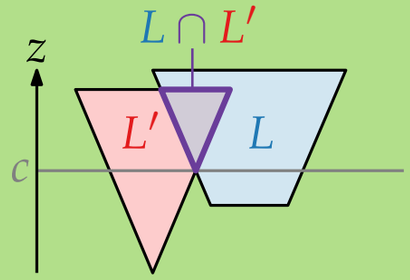
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



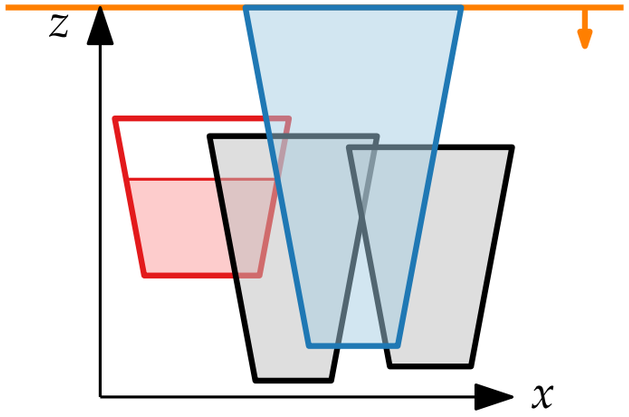

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



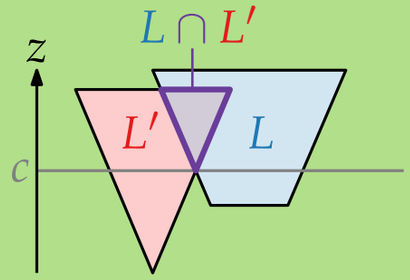
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



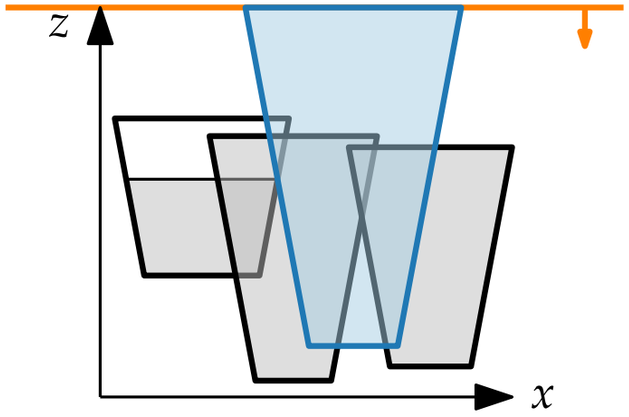

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



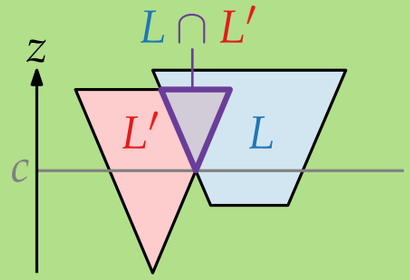
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



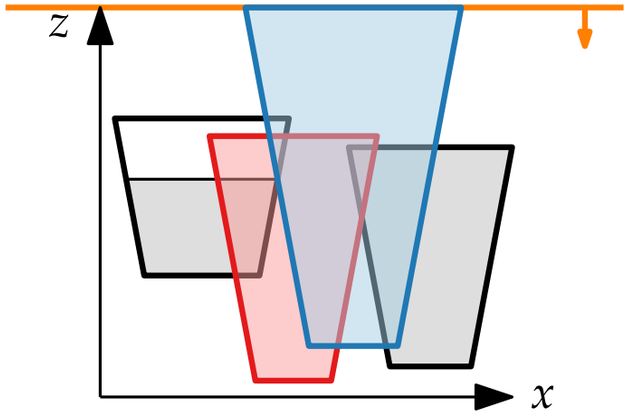

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



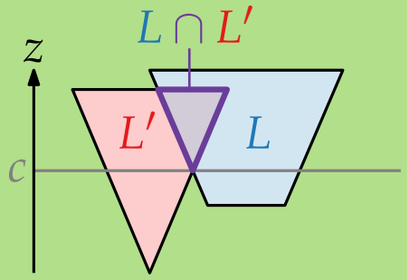
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



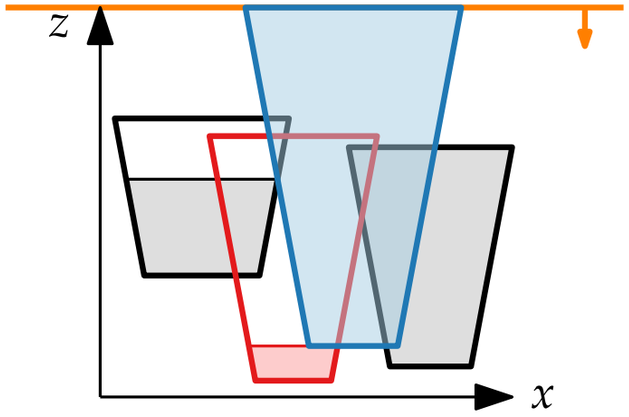

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
```



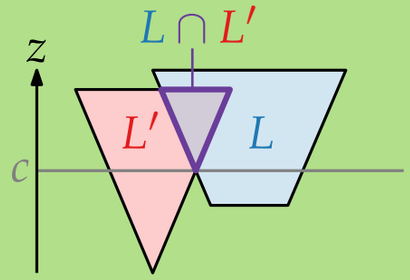
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



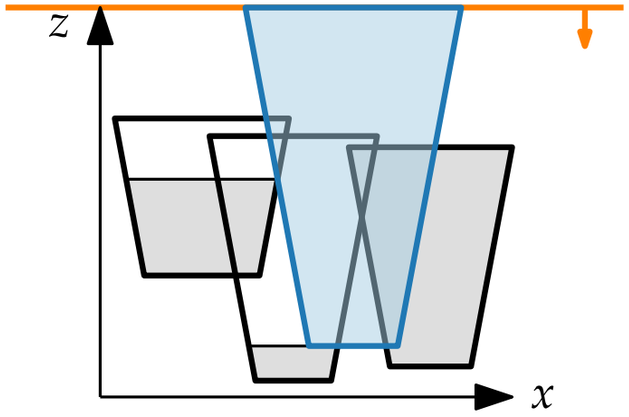

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



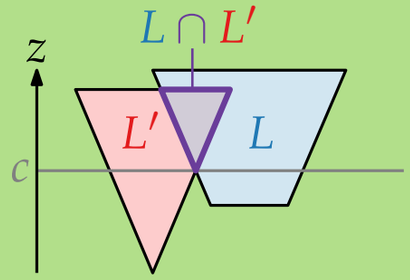
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



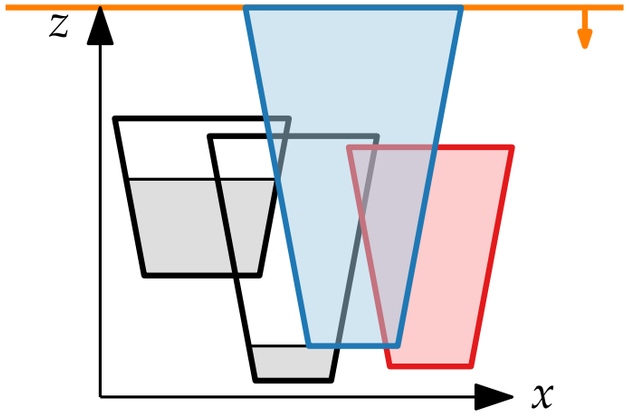

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



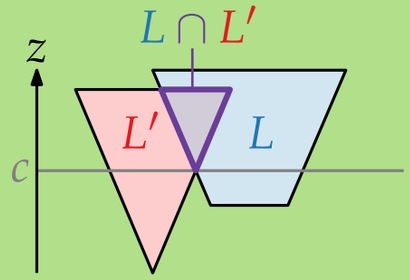
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



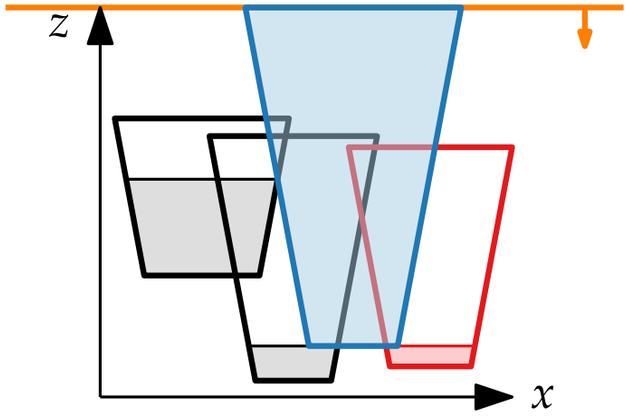

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



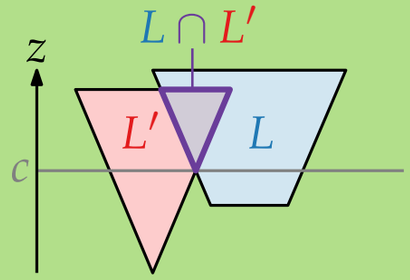
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



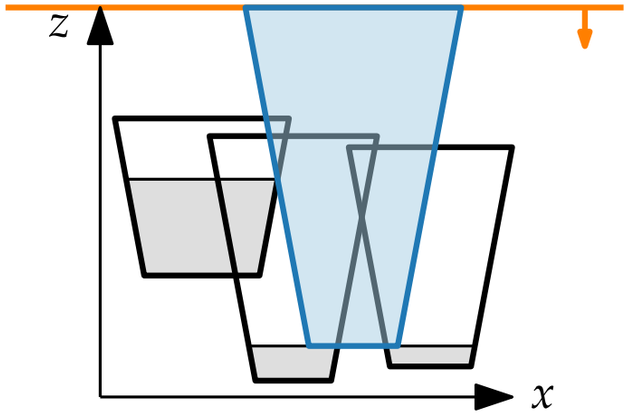

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



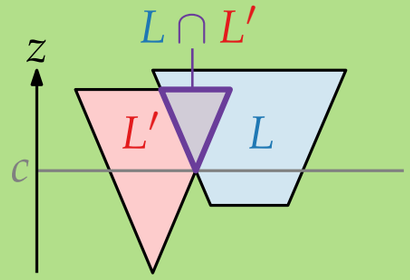
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



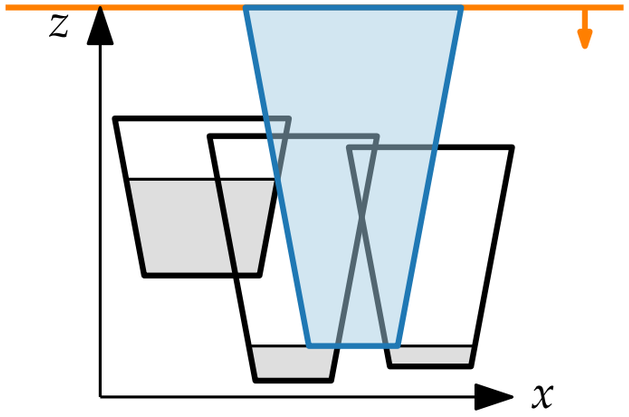

```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
```



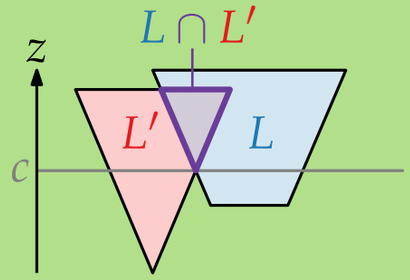
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$




```
GreedyTopDown(L)
    foreach L ∈ L do
        (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
    Q = PriorityQueue(L)
    while !Q.isEmpty() do
        L = Q.extractMax()
        foreach L' ∈ Q do
            if L ∩ L' ≠ ∅ then
                c = kleinste z-Koordinate von L ∩ L'
                A_L' = min{A_L', c}
                if A_L' = a_L' then Q.remove(L')
```

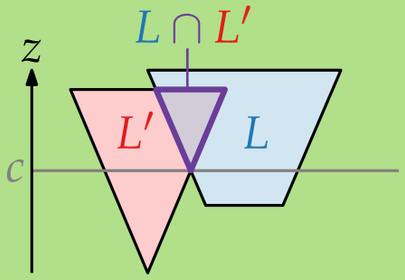


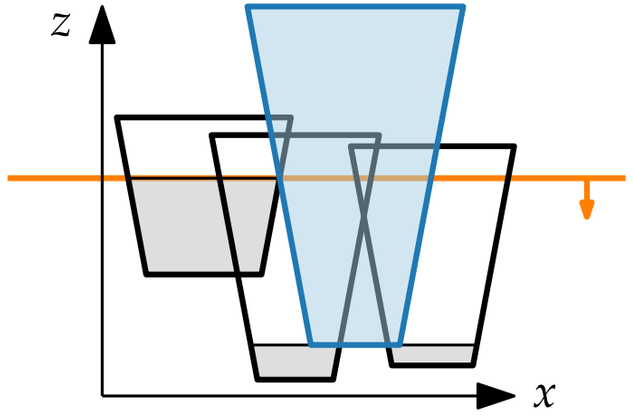
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```

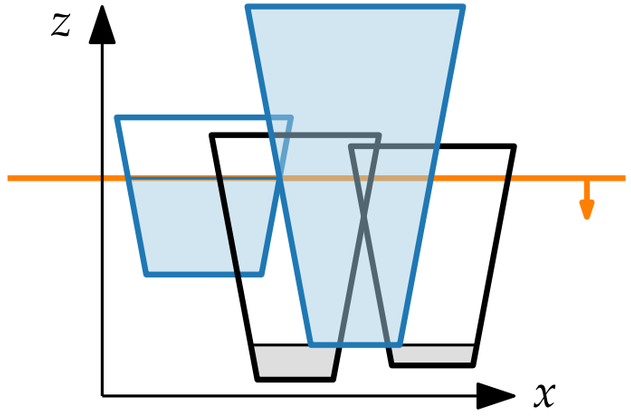




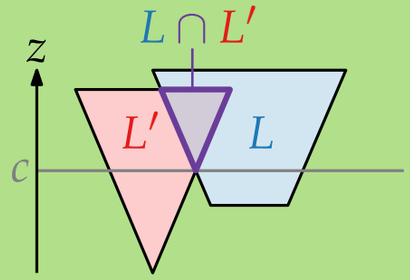
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



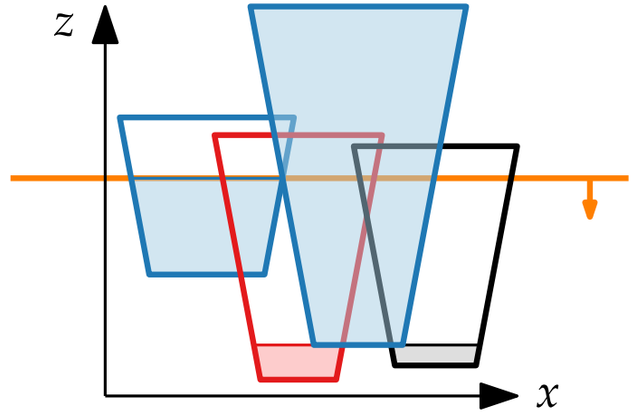
GreedyTopDown($\mathcal{L}$)
- **foreach** $L \in \mathcal{L}$ **do**
  - $(a_L, A_L) = (s_L, S_L)$ (absteigend sortiert nach (A_L, S_L))
- Q = PriorityQueue($\mathcal{L}$)
- **while** !Q.isEmpty() **do**
  - $L = Q$.extractMax()
  - **foreach** $L' \in Q$ **do**
    - **if** $L \cap L' \neq \emptyset$ **then**
      - c = kleinste z-Koordinate von $L \cap L'$
      - $A_{L'} = \min\{A_{L'}, c\}$
      - **if** $A_{L'} = a_{L'}$ **then** Q.remove(L')



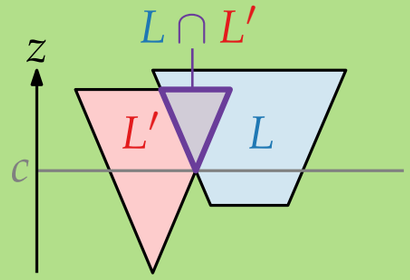
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$



GreedyTopDown($\mathcal{L}$)
- **foreach** $L \in \mathcal{L}$ **do**
  - $(a_L, A_L) = (s_L, S_L)$ (absteigend sortiert nach (A_L, S_L))
- Q = PriorityQueue($\mathcal{L}$)
- **while** !Q.isEmpty() **do**
  - $L = Q$.extractMax()
  - **foreach** $L' \in Q$ **do**
    - **if** $L \cap L' \neq \emptyset$ **then**
      - c = kleinste z-Koordinate von $L \cap L'$
      - $A_{L'} = \min\{A_{L'}, c\}$
      - **if** $A_{L'} = a_{L'}$ **then** Q.remove(L')

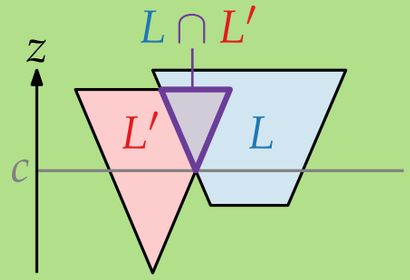


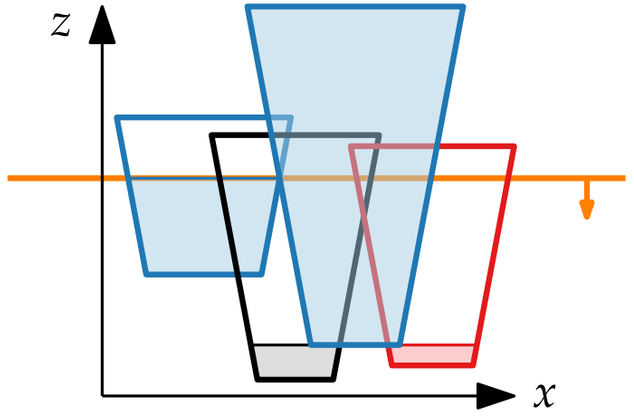
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```

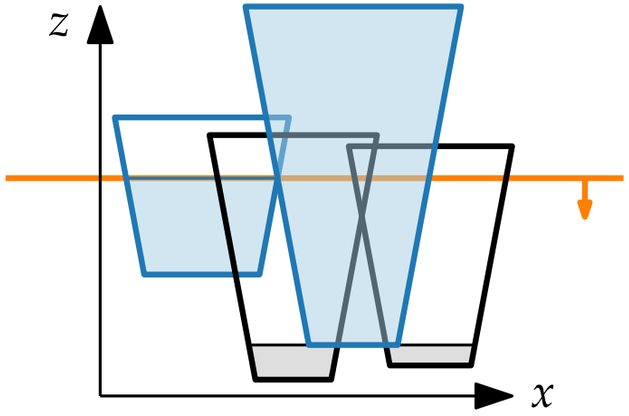




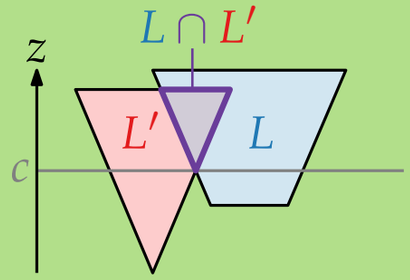
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$




```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```

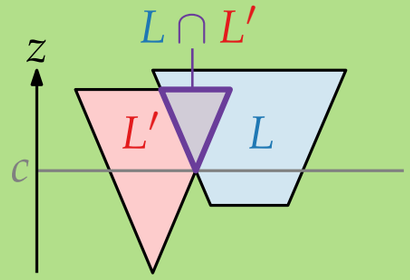


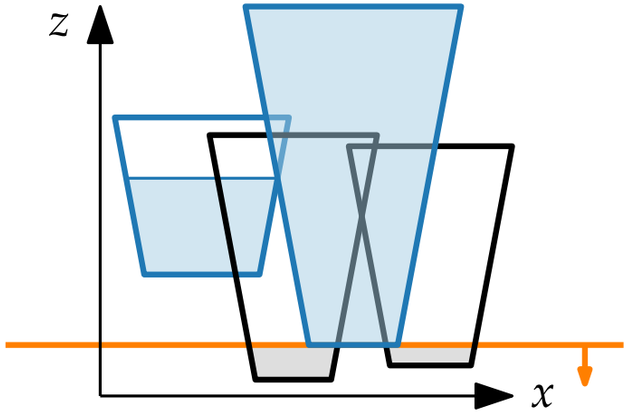
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



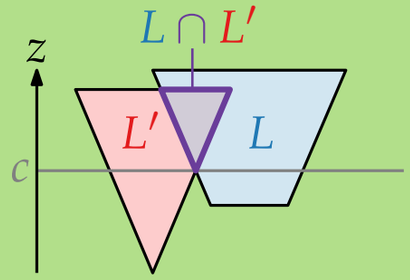


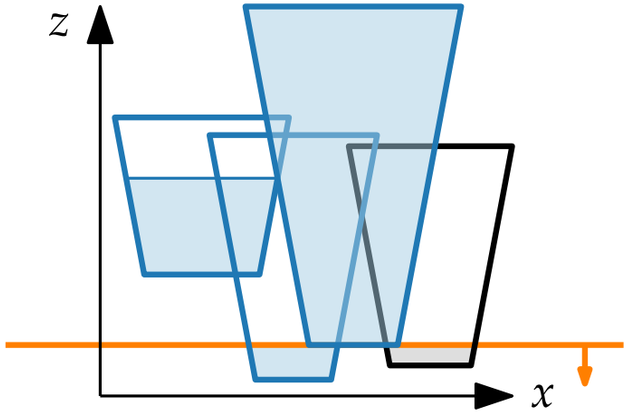
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



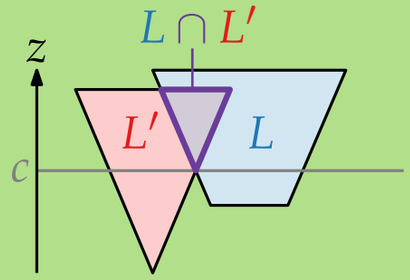


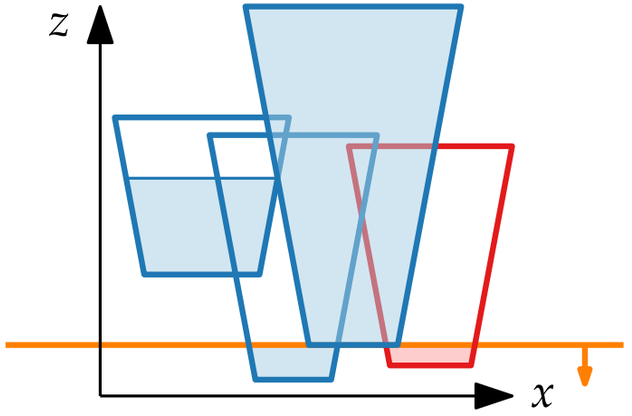
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



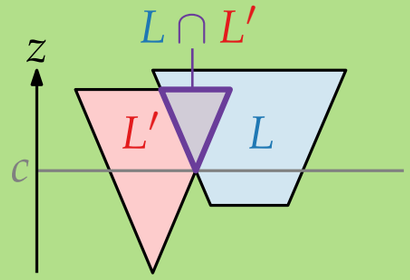


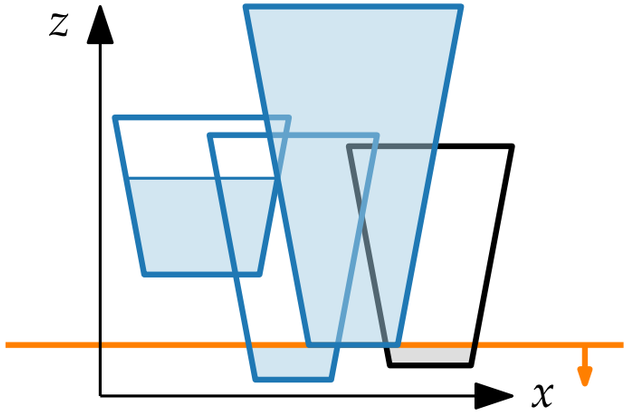
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



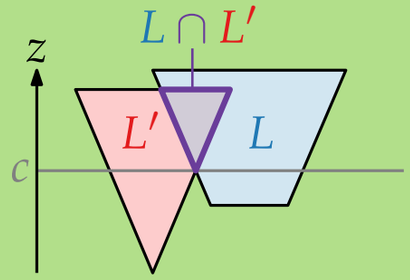


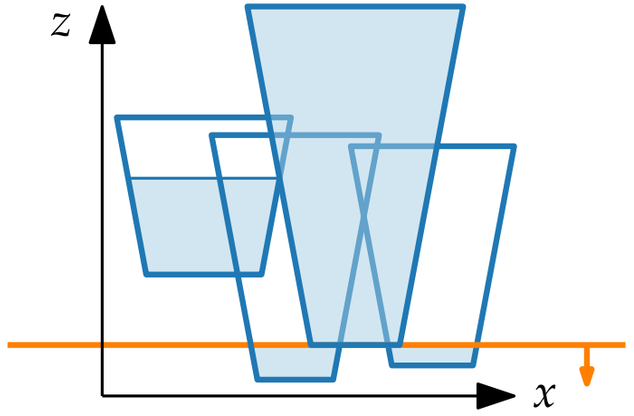
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



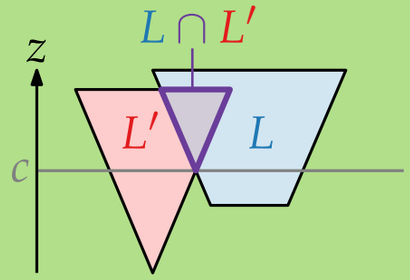


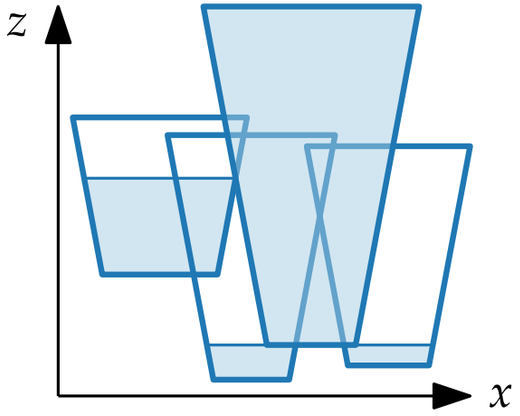
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



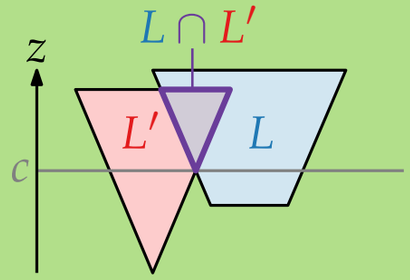


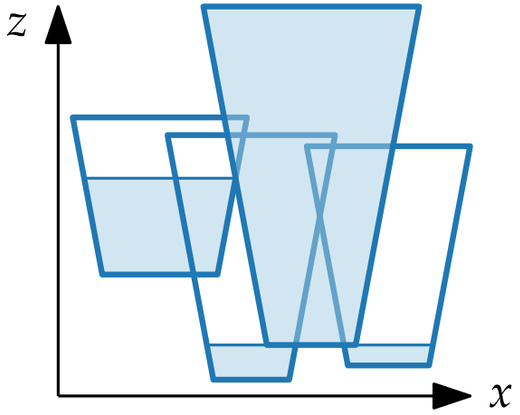
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```





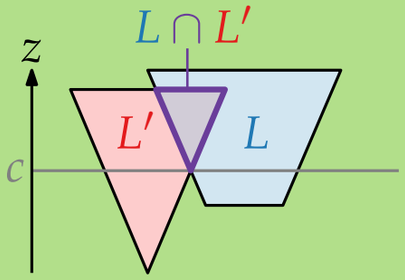
optimal?

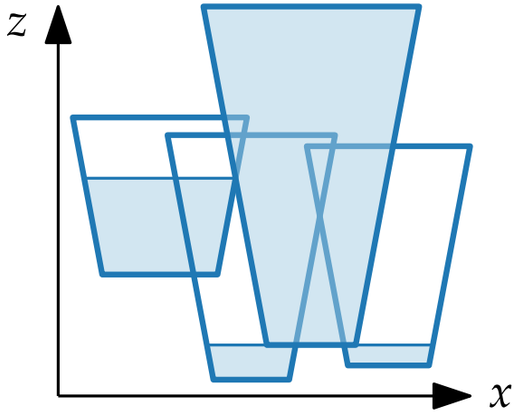
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

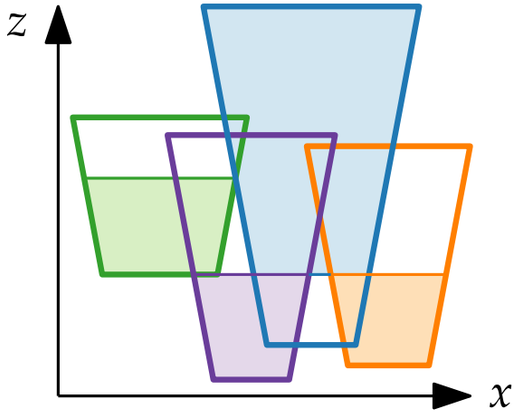
Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```





optimal?

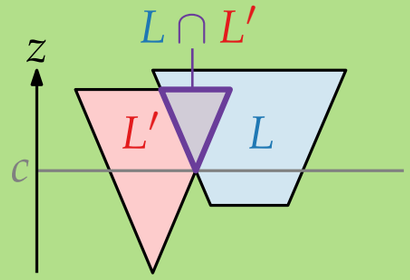


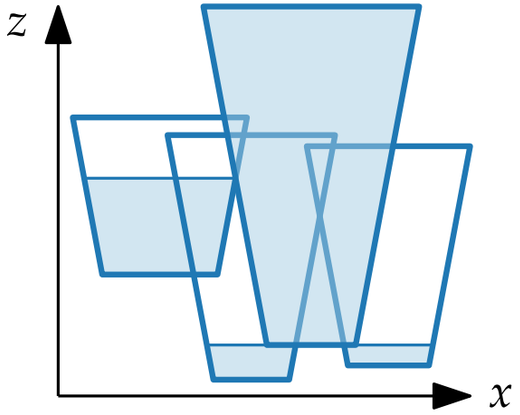
Greedy Top-Down Sweep Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

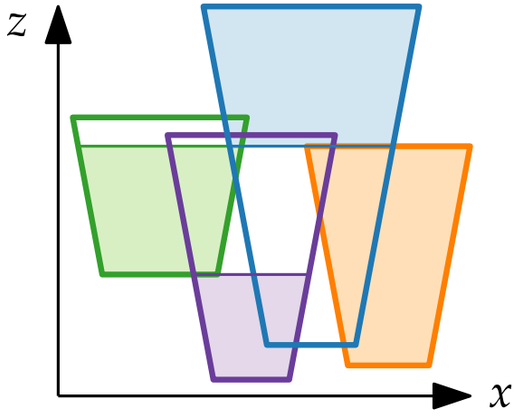
Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

```
GreedyTopDown(L)
  foreach L ∈ L do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(L)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```





optimal?

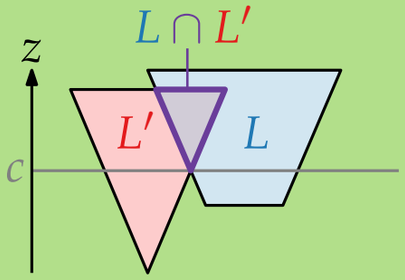


Greedy Top-Down Sweep Algorithmus

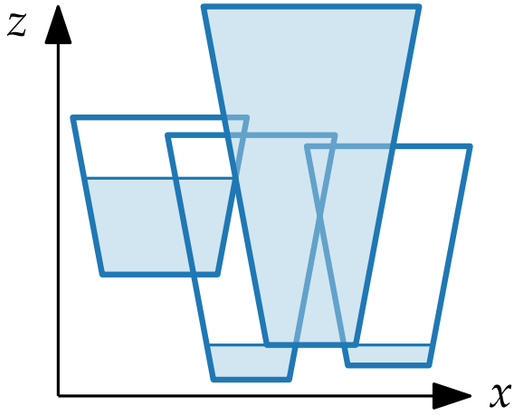
Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

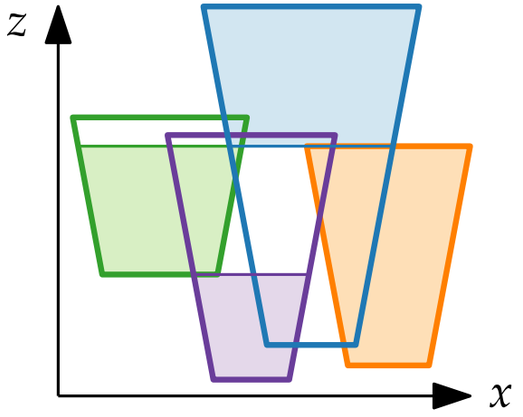
```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



Laufzeit?



optimal?

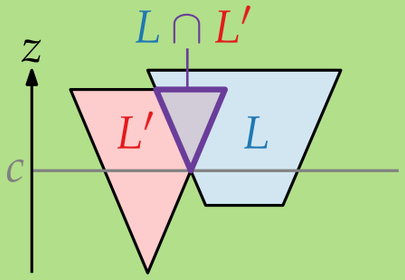


Greedy Top-Down Sweep Algorithmus

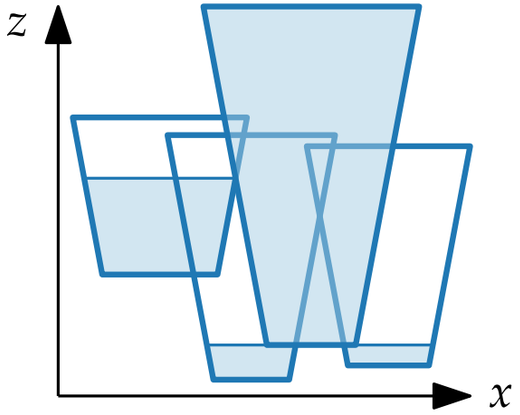
Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

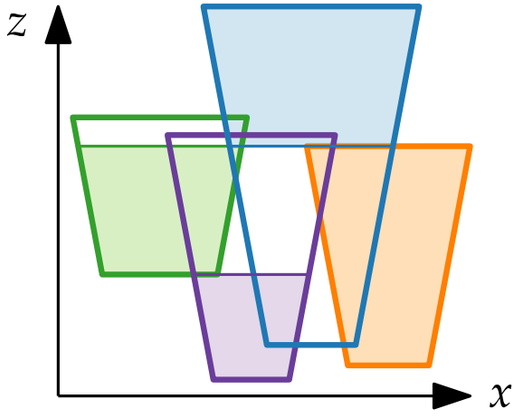
```
GreedyTopDown(ℒ)
  foreach L ∈ ℒ do
    (a_L, A_L) = (s_L, S_L)        // absteigend sortiert nach (A_L, S_L)
  Q = PriorityQueue(ℒ)
  while !Q.isEmpty() do
    L = Q.extractMax()
    foreach L' ∈ Q do
      if L ∩ L' ≠ ∅ then
        c = kleinste z-Koordinate von L ∩ L'
        A_L' = min{A_L', c}
        if A_L' = a_L' then Q.remove(L')
```



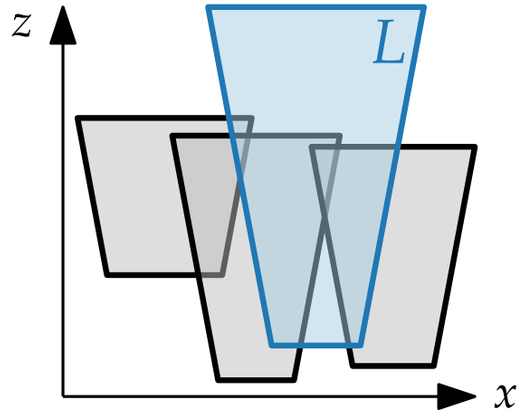
Laufzeit?
$\mathcal{O}(n^2)$



optimal?

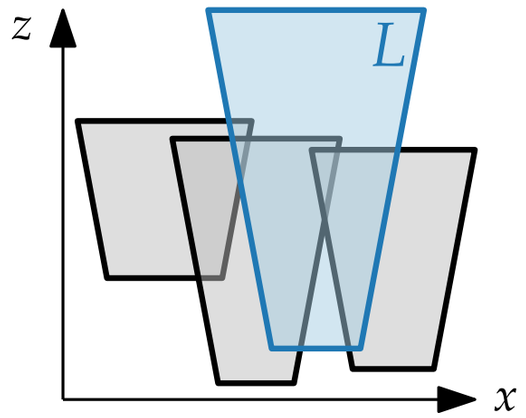


Blockade-Lemma



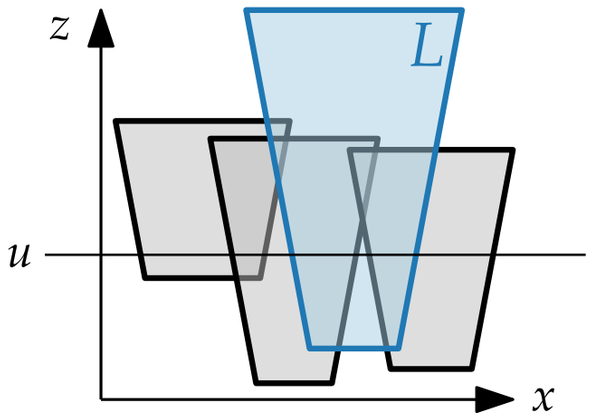
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$



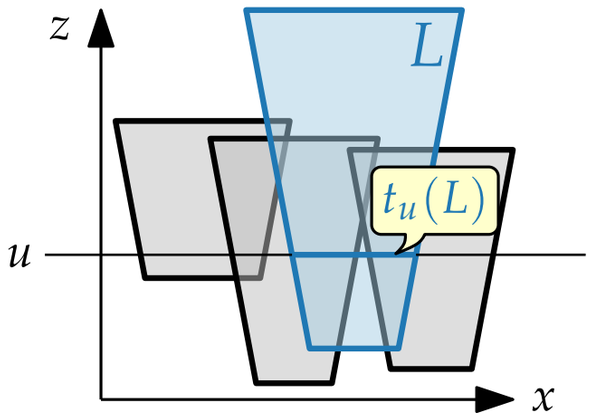
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$



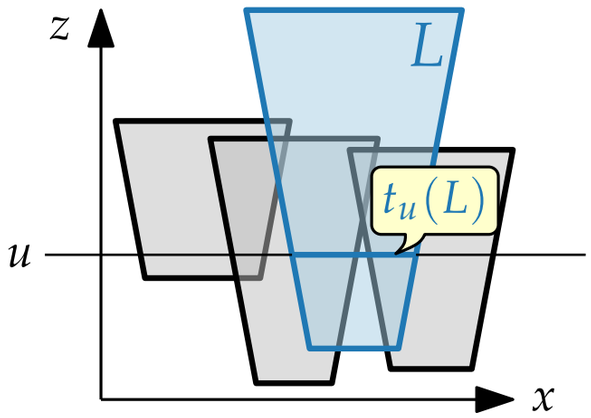
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$



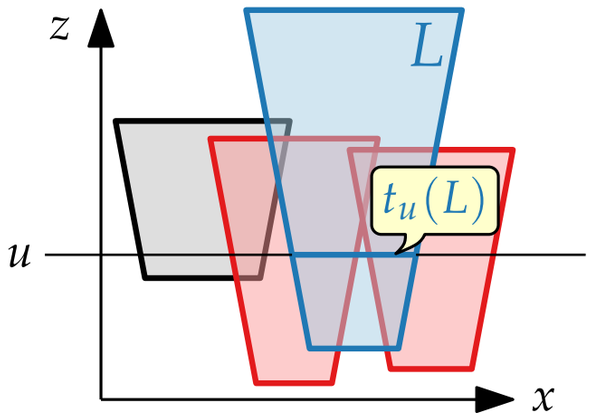
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$



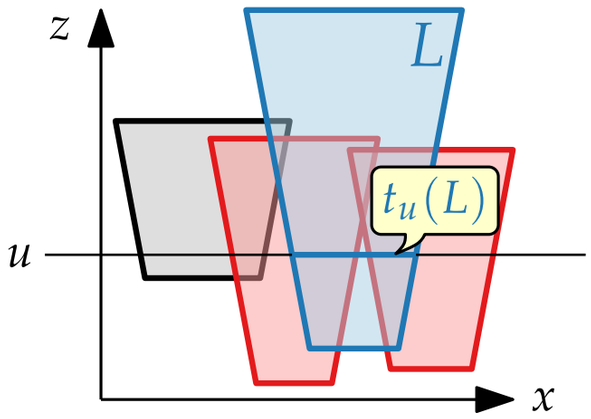
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$



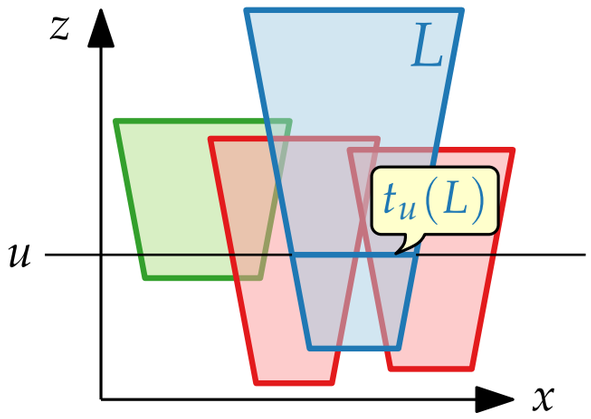
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



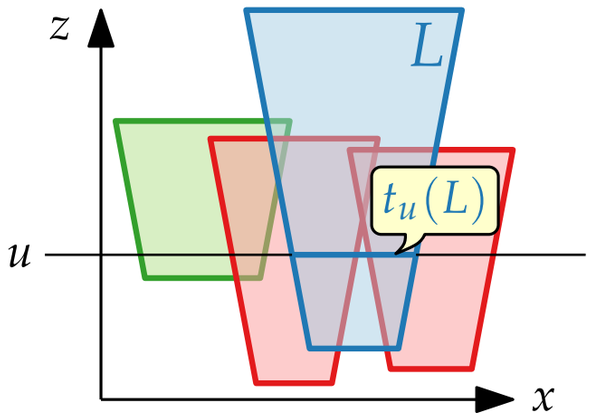
Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$

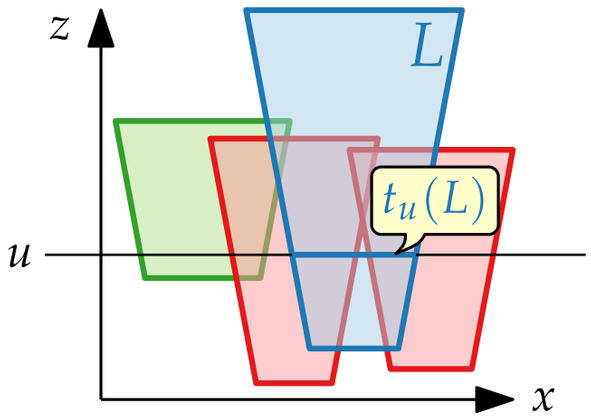


Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



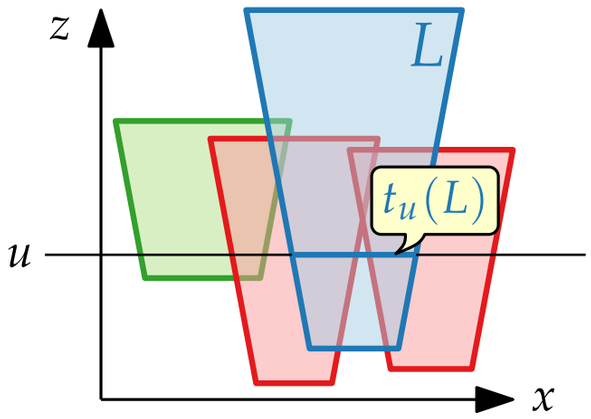
Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Beweis.

Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



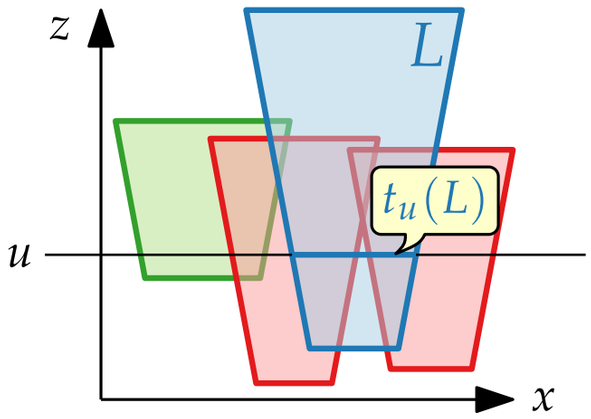
Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Beweis. Sei L ein beliebiges Label mit $a_L \leq u \leq A_L$.

Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



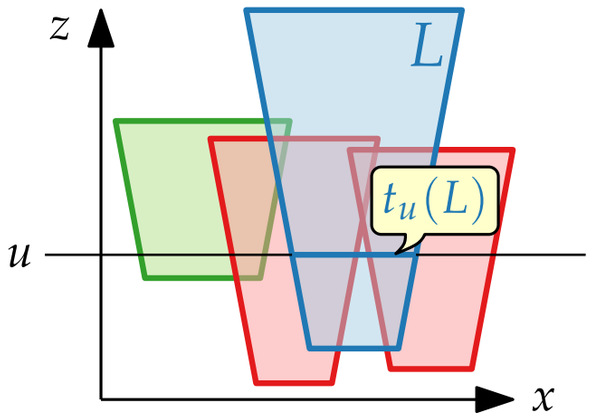
Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Beweis. Sei L ein beliebiges Label mit $a_L \leq u \leq A_L$.
Betrachte eine optimale Lösung.

Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$

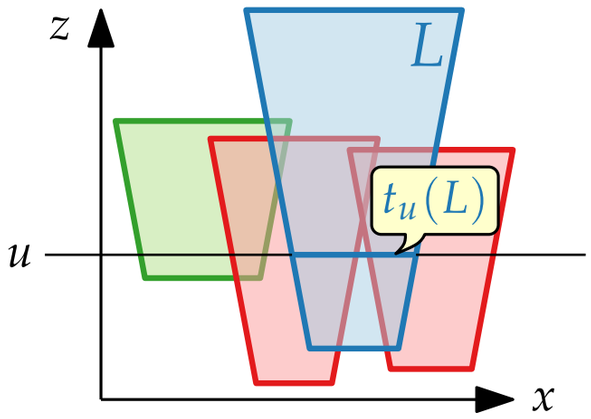


Blockade-Lemma.
Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Beweis. Sei L ein beliebiges Label mit $a_L \leq u \leq A_L$.
Betrachte eine optimale Lösung.
Wenn sie L für z-Wert u nicht wählt, dann maximal c andere Labels.

Blockade-Lemma

- die **Spur** $t_u(L)$ eines Labelkörpers L zum Wert u ist der Schnitt von L mit der Ebene $\pi(u) : z = u$
- L **blockiert** L' an Wert u in Lösung $\mathcal{A}$, wenn $a_L \leq u \leq A_L$ und $t_u(L) \cap t_u(L') \neq \emptyset$
- L und L' sind **unabhängig** am Wert u, falls $t_u(L) \cap t_u(L') = \emptyset$



Blockade-Lemma.
Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Beweis. Sei L ein beliebiges Label mit $a_L \leq u \leq A_L$.
Betrachte eine optimale Lösung.
Wenn sie L für z-Wert u nicht wählt, dann maximal c andere Labels. □

Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.

Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.

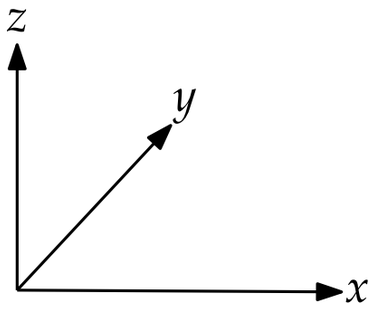
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

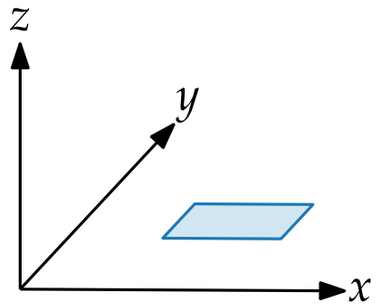
Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.



Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.



Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.

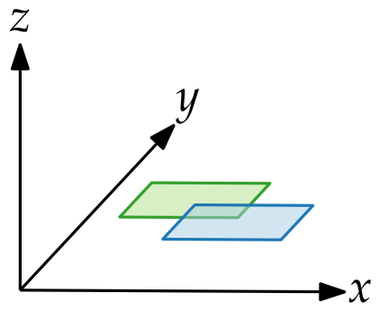
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.



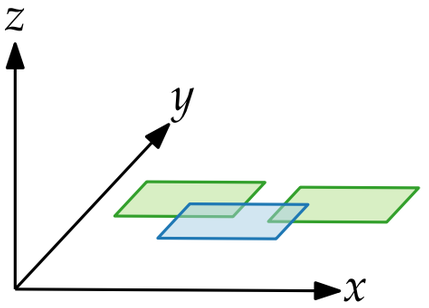
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.



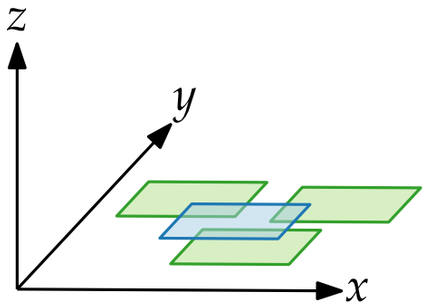
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.



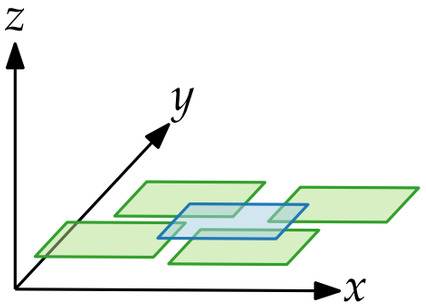
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine -Approximation.



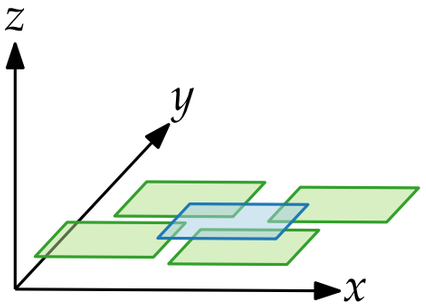
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



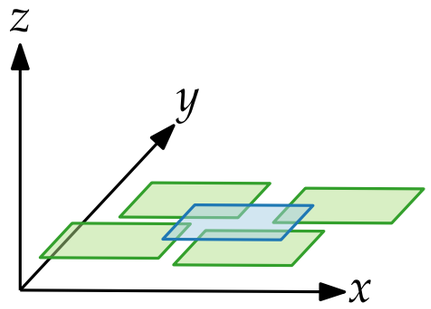
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.

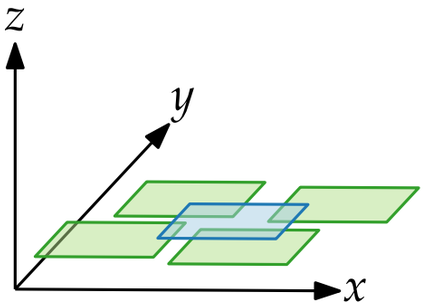
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

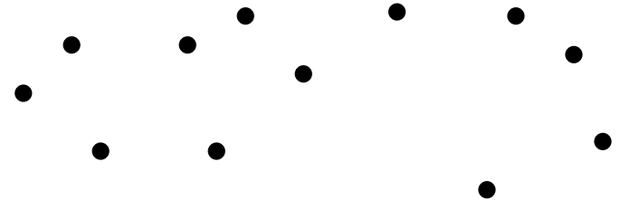
Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.



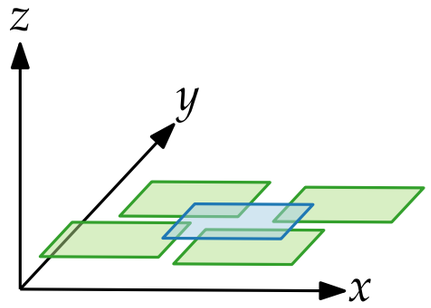
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

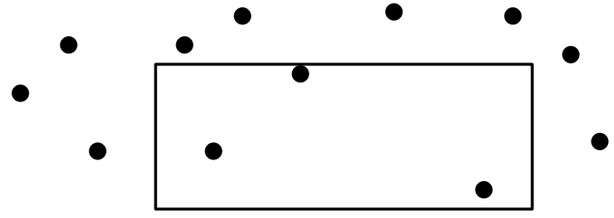
Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.



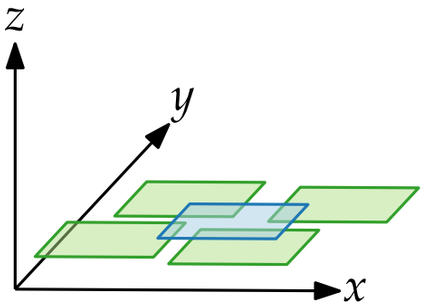
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

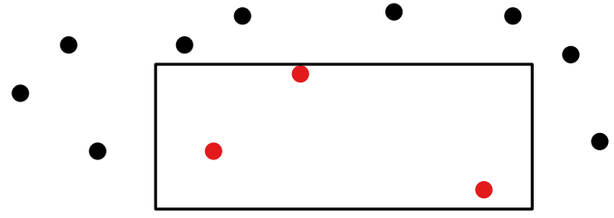
Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.



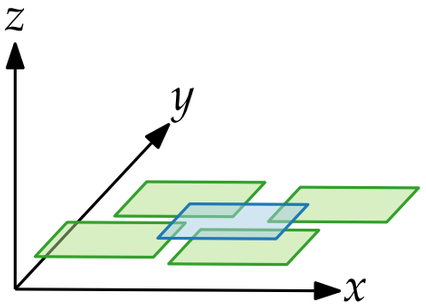
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

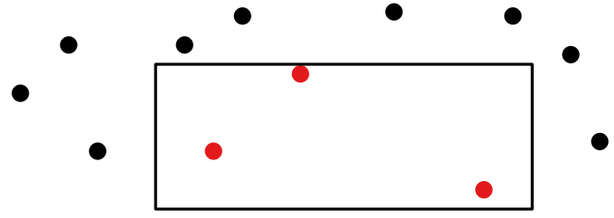
Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.

Aufbau: $\mathcal{O}(n \log n)$, Speicher: $\mathcal{O}(n \log n)$, Query: $\mathcal{O}(k + \log^2 n)$



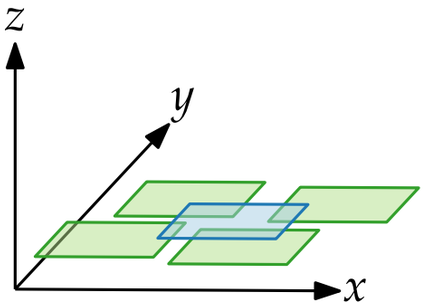
Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.

Satz.

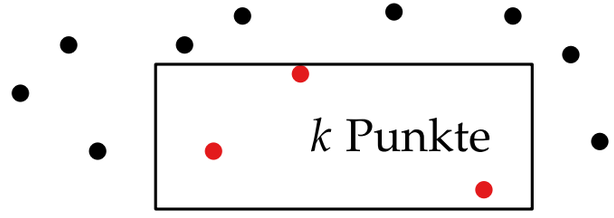
Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.



Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.

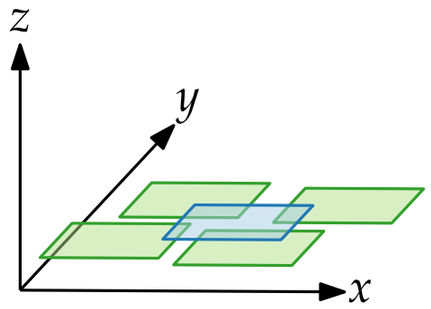
Aufbau: $\mathcal{O}(n \log n)$, Speicher: $\mathcal{O}(n \log n)$, Query: $\mathcal{O}(k + \log^2 n)$



Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.



Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.

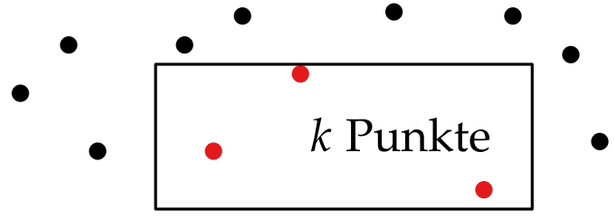
Lemma.

GreedyTopDown lässt sich in $\mathcal{O}((n+k)\log^2 n)$ Zeit implementieren.

Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.

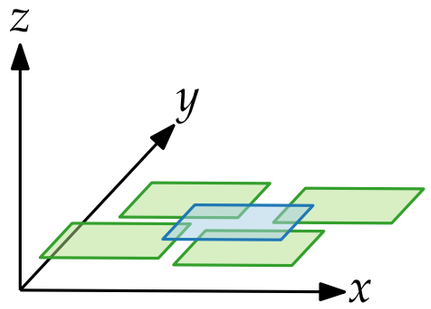
Aufbau: $\mathcal{O}(n \log n)$, Speicher: $\mathcal{O}(n \log n)$, Query: $\mathcal{O}(k + \log^2 n)$



Approximation für Einheitslabel

Blockade-Lemma.

Sei $\mathcal{L}$ eine Menge von Labelkörpern. Falls jedes $L \in \mathcal{L}$ für jeden z-Wert u nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert GreedyTopDown eine $(1/c)$-Approximation.



Satz.

Für n achsenparallele Labels gleicher Größe berechnet GreedyTopDown eine $(1/4)$-Approximation.

Lemma.

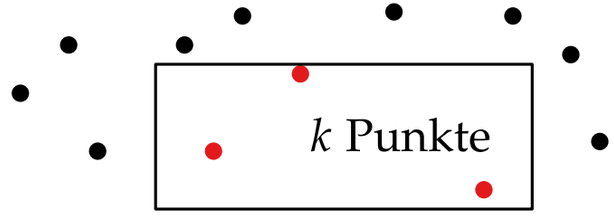
Anzahl von Seiten-Schnitten der Labelpyramiden

GreedyTopDown lässt sich in $\mathcal{O}((n+k)\log^2 n)$ Zeit implementieren.

Range Tree:

Geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Menge P von n Punkten.

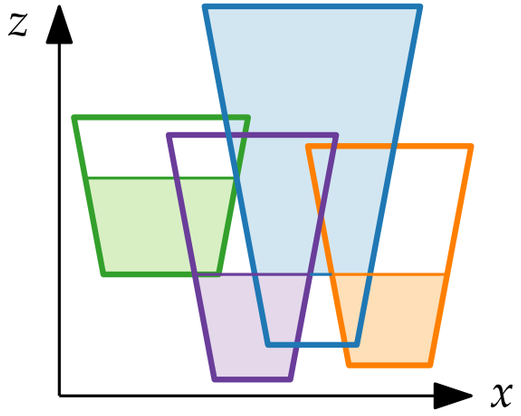
Aufbau: $\mathcal{O}(n \log n)$, Speicher: $\mathcal{O}(n \log n)$, Query: $\mathcal{O}(k + \log^2 n)$



Universität Trier

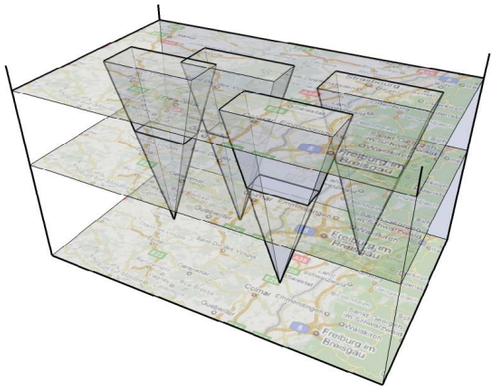
Algorithmen für geographische Informationssysteme

7. Vorlesung
Dynamische Kartenbeschriftung: Zoomen



Teil IV:
Ebenenbasierter Greedy-Algorithmus

nicht prüfungsrelevant



Philipp Kindermann

Wintersemester 2024/25

Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle (s_L, S_L) für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle (a_L, A_L) für alle $L \in \mathcal{L}$

Ebenenbasierter Greedy-Algorithmus

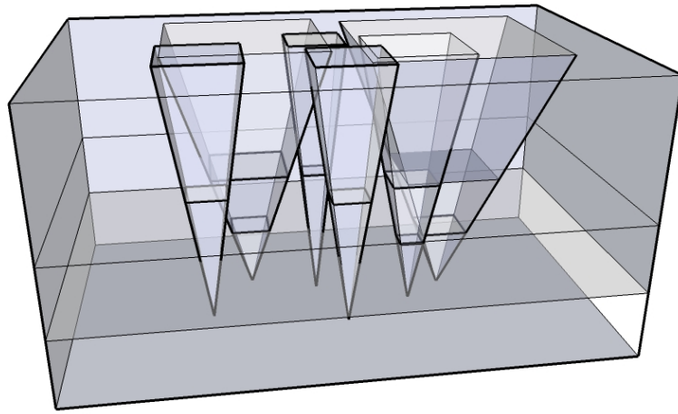
Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

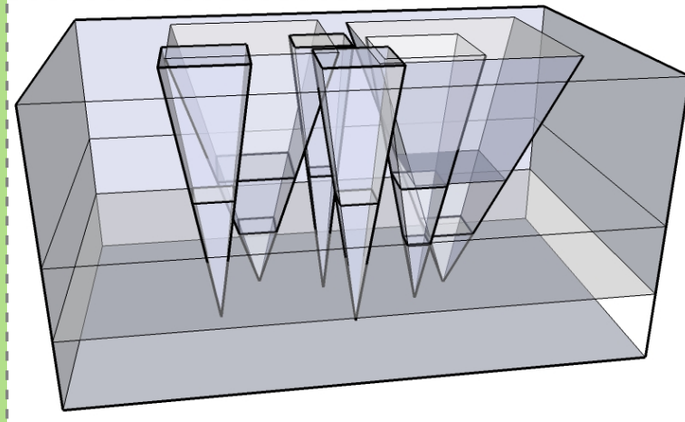


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)

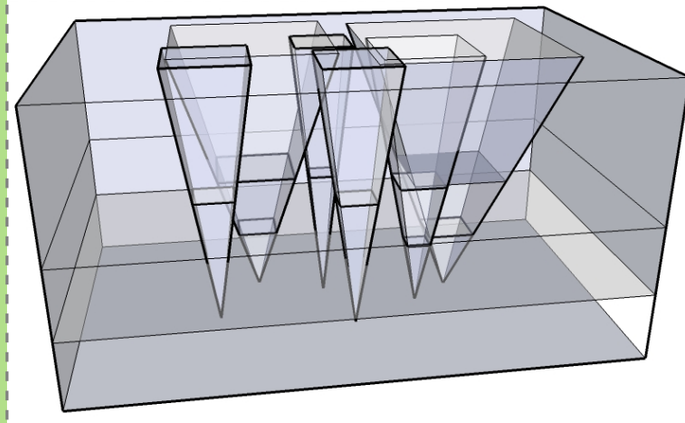


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $L.\text{active} = \text{false}$; $A_L = 0$

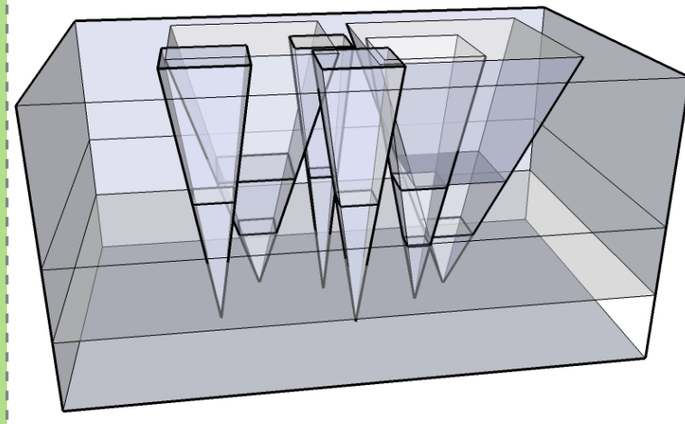


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)
  foreach $L \in \mathcal{L}$ **do**
    $L.\text{active} = \text{false}$; $A_L = 0$
  for $i = 0$ **to** $\log_2 n$ **do**

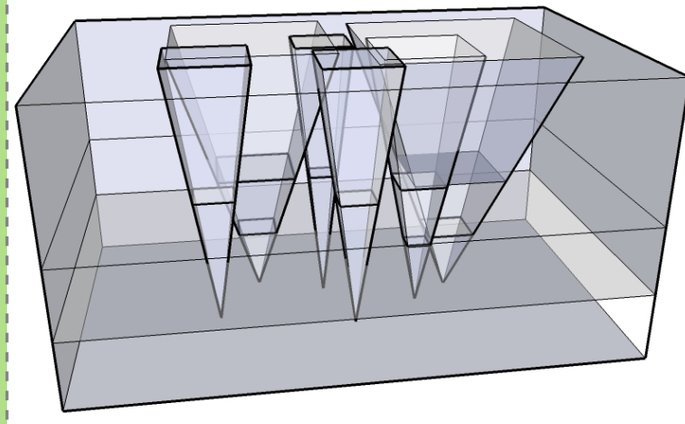


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                    // Phase i
```

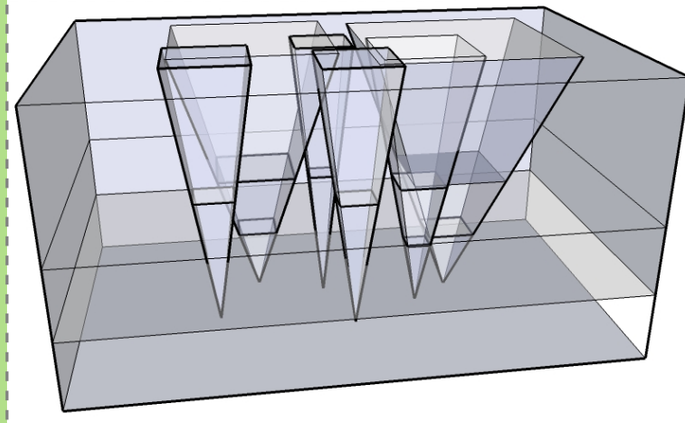


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                    // Phase i
    u = S_max / 2^i; π_i = π(u)
```

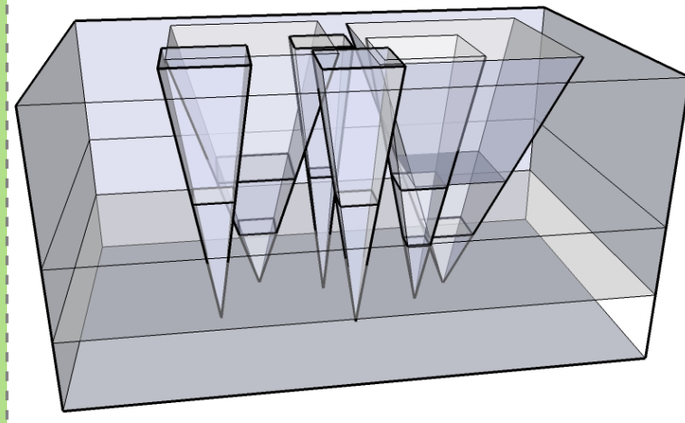


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
```

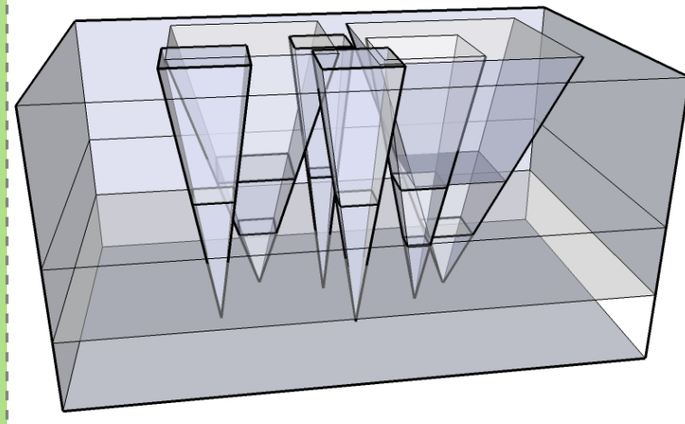


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)
- **foreach** $L \in \mathcal{L}$ **do**
  - $L.\text{active} = \text{false}$; $A_L = 0$
- **for** $i = 0$ **to** $\log_2 n$ **do** // Phase i
  - $u = S_{\max}/2^i$; $\pi_i = \pi(u)$ // betrachtete z-Koordinate
  - $\mathcal{T}$ = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden

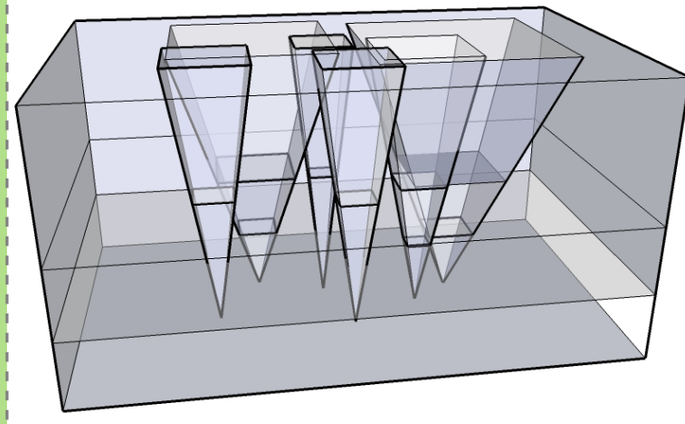


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
```

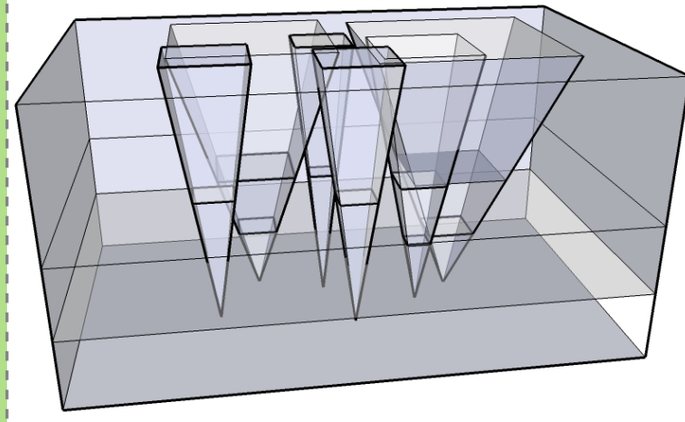


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)
- **foreach** $L \in \mathcal{L}$ **do**
  - $L.\text{active} = \text{false}$; $A_L = 0$
- **for** $i = 0$ **to** $\log_2 n$ **do** // Phase i
  - $u = S_{\max}/2^i$; $\pi_i = \pi(u)$ // betrachtete z-Koordinate
  - $\mathcal{T} =$ Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
  - **while** $\mathcal{T} \neq \emptyset$ **do**
    - $L =$ Labelpyramide in $\mathcal{T}$ mit kleinster Spur

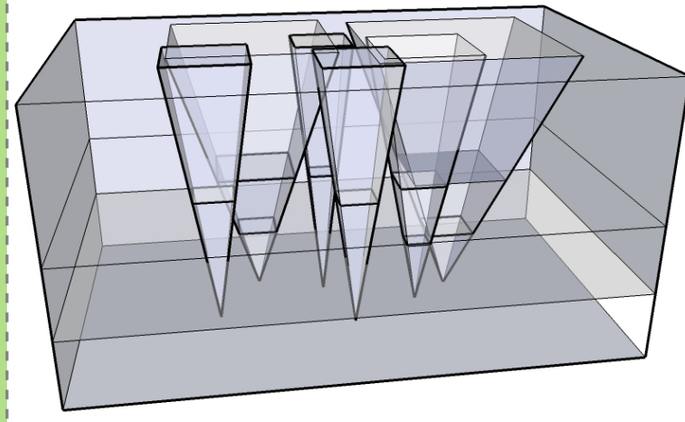


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
```

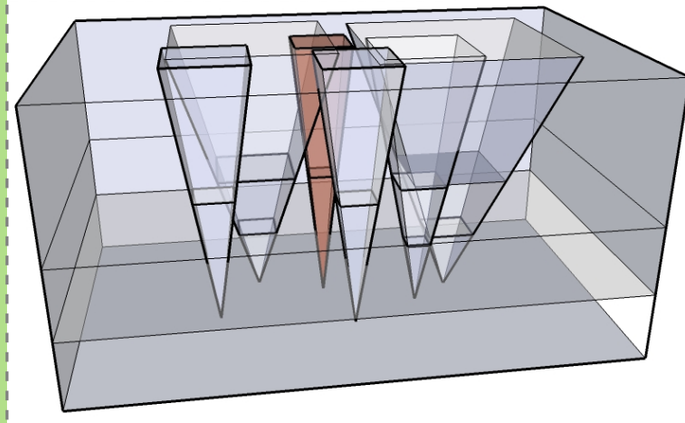


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
```

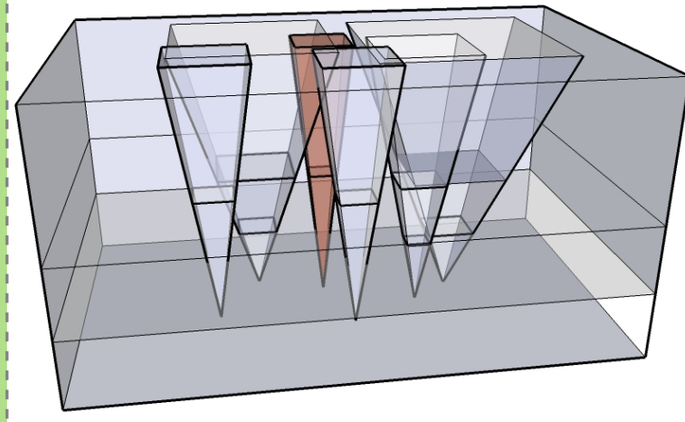


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
```

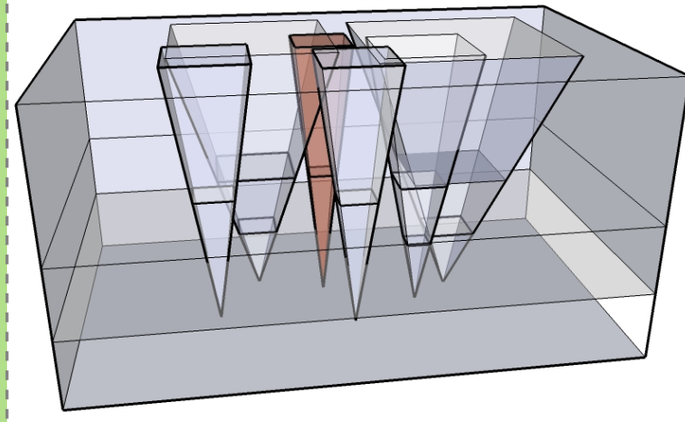


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
      Entferne L und alle in π_i geschnittenen Labels aus T
```

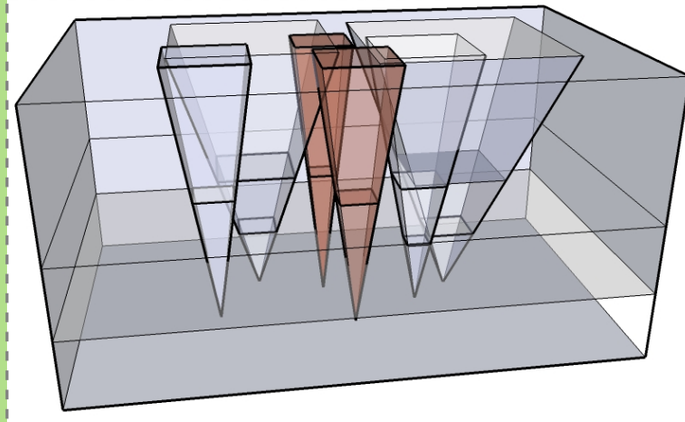


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
      Entferne L und alle in π_i geschnittenen Labels aus T
```

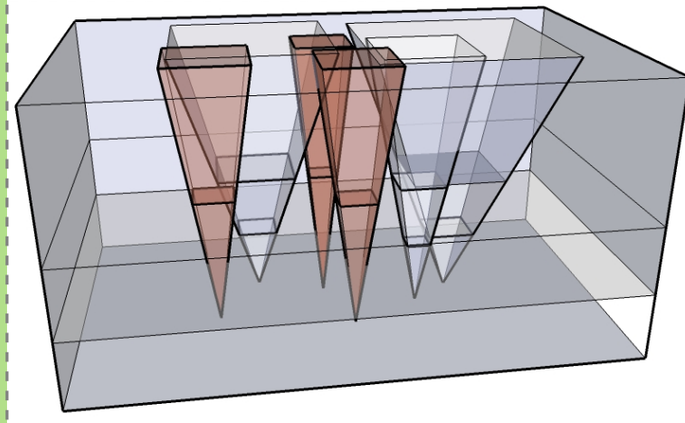


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
      Entferne L und alle in π_i geschnittenen Labels aus T
```

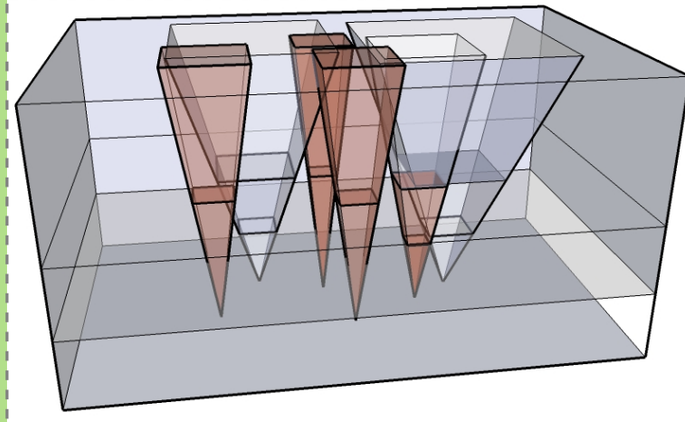


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

LayeredTopDown($\mathcal{L}$)
- **foreach** $L \in \mathcal{L}$ **do**
  - $L.\text{active} = \text{false}$; $A_L = 0$
- **for** $i = 0$ **to** $\log_2 n$ **do** // Phase i
  - $u = S_{\max}/2^i$; $\pi_i = \pi(u)$ // betrachtete z-Koordinate
  - $\mathcal{T}$ = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
  - **while** $\mathcal{T} \neq \emptyset$ **do**
    - L = Labelpyramide in $\mathcal{T}$ mit kleinster Spur
    - $L.\text{active} = \text{true}$
    - $A_L = S$
    - Entferne L und alle in π_i geschnittenen Labels aus $\mathcal{T}$

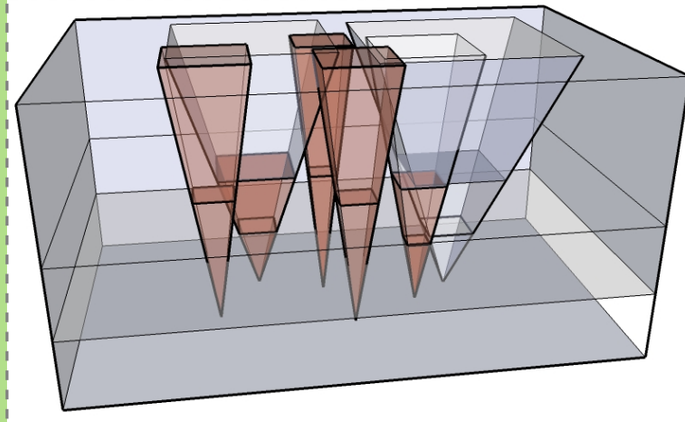


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
      Entferne L und alle in π_i geschnittenen Labels aus T
```

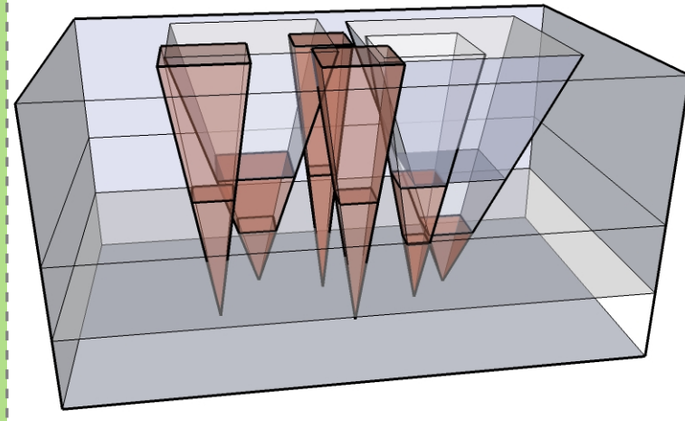


Ebenenbasierter Greedy-Algorithmus

Gegeben:
- Menge $\mathcal{L}$ von (offenen) quadratischen Labelpyramiden
- Verfügbare Intervalle $(0, S_{\max})$ für alle $L \in \mathcal{L}$

Gegeben: Aktive Intervalle $(0, A_L)$ für alle $L \in \mathcal{L}$

```
LayeredTopDown(L)
  foreach L ∈ L do
    L.active = false; A_L = 0
  for i = 0 to log_2 n do                                  // Phase i
    u = S_max/2^i; π_i = π(u)                  // betrachtete z-Koordinate
    T = Inaktive Labelpyramiden, die keine aktiven Labels in π_i schneiden
    while T ≠ ∅ do
      L = Labelpyramide in T mit kleinster Spur
      L.active = true
      A_L = S
      Entferne L und alle in π_i geschnittenen Labels aus T
```



Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert. Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.

Sei L ein aktives Label in Ebene π_i.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.
B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.

➡ L war in Phase h schon aktiv (sonst A)

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

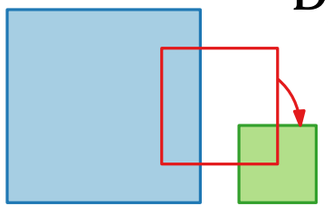
B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.

➡ L war in Phase h schon aktiv (sonst A)



Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

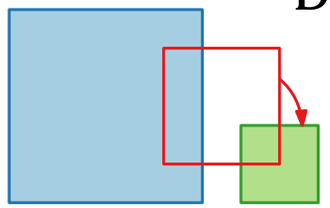
A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.



➡ L war in Phase h schon aktiv (sonst A)

Angenommen $\ell' < \ell/3$

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

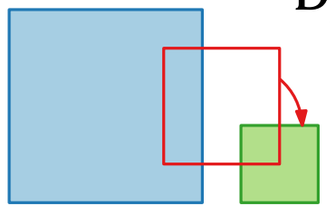
A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.



➡ L war in Phase h schon aktiv (sonst A)

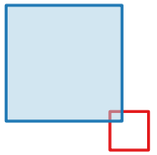
Angenommen $\ell' < \ell/3$ ➡ L enthält L' komplett in allen Ebenen $\pi_{\leq i-1}$.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

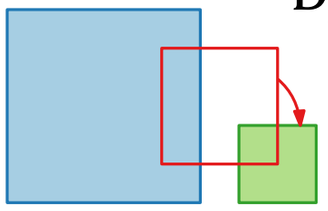
B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.



Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.



➡ L war in Phase h schon aktiv (sonst A)

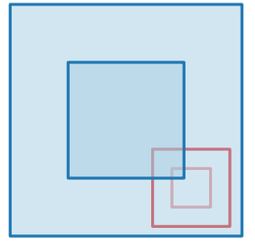
Angenommen $\ell' < \ell/3$ ➡ L enthält L' komplett in allen Ebenen $\pi_{\leq i-1}$.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

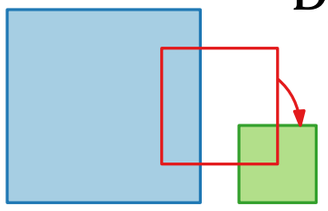
B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.



Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.



➡ L war in Phase h schon aktiv (sonst A)

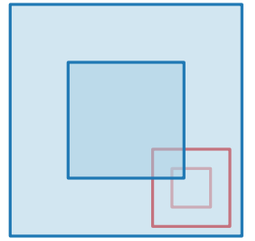
Angenommen $\ell' < \ell/3$ ➡ L enthält L' komplett in allen Ebenen $\pi_{\leq i-1}$.

Blocker-Zuordnung

Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

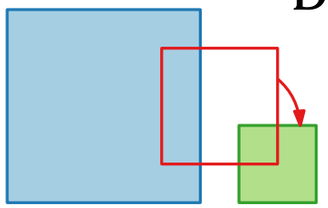
B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.



Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.



➡ L war in Phase h schon aktiv (sonst A)

Angenommen $\ell' < \ell/3$ ➡ L enthält L' komplett in allen Ebenen $\pi_{\leq i-1}$.

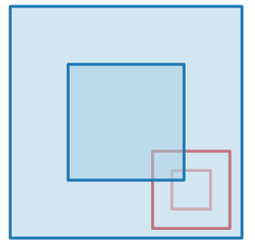
➡ L schneidet L'' in Ebene π_h.

Blocker-Zuordnung

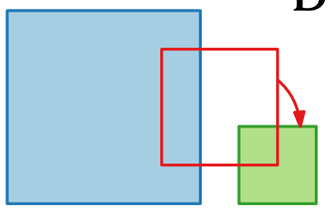
Am Ende von Phase i werden alle inaktiven Labels durch ein aktives Label blockiert.
Sei L' ein blockiertes Label. Wir weisen L' einem aktiven Label L zu, da es blockiert.

A) Falls L' anfangs in $\mathcal{T}$ war, wurde L' durch neu aktiviertes Label L blockiert. Weise L' dem Label L zu.

B) Sonst weise L' einem beliebigenden blockierenden Label L zu, das schon zu Beginn der Phase i aktiv war.



Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Beweis. A) L wurde vor L' in Phase i ausgewählt $\Rightarrow$ L' hat Seitenlänge $\geq \ell$.

B) Sei $h < i$ die letzte Phase, in der L' nicht L sondern L'' zugewiesen wurde.

➡ L war in Phase h schon aktiv (sonst A)

Angenommen $\ell' < \ell/3$ ➡ L enthält L' komplett in allen Ebenen $\pi_{\leq i-1}$.

➡ L schneidet L'' in Ebene π_h. ↯

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine -Approximation.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i.
Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Abschätzung aktive Gesamthöhe

Lemma.

Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.

Abschätzung aktive Gesamthöhe

Lemma.

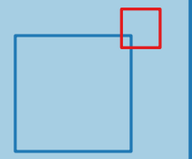
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

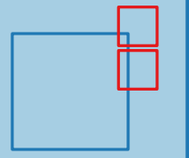
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

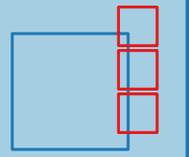
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

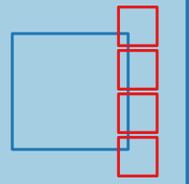
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

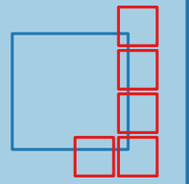
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

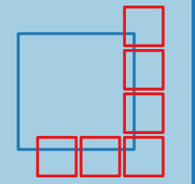
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

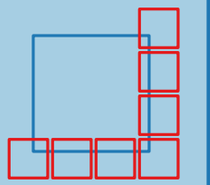
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

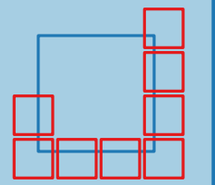
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

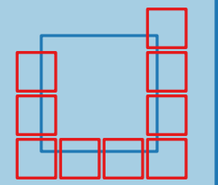
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

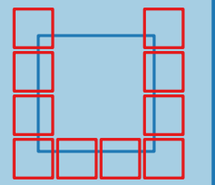
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

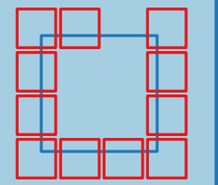
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

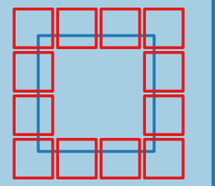
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

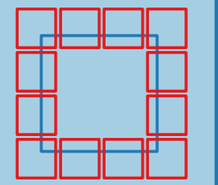
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



Abschätzung aktive Gesamthöhe

Lemma.

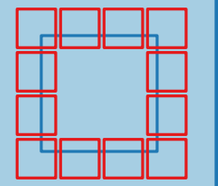
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



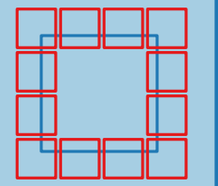
➡ $c \leq 12$

Abschätzung aktive Gesamthöhe

Lemma.
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.
- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz.
Für n achsenparallele Quadrate berechnet LayeredGreedy eine -Approximation in $\mathcal{O}(n \log^3 n)$ Zeit.

Abschätzung aktive Gesamthöhe

Lemma.

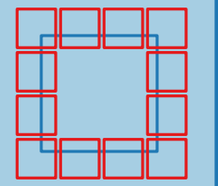
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz.

Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit.

Abschätzung aktive Gesamthöhe

Lemma.

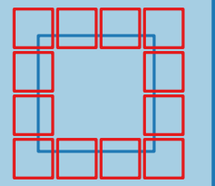
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz.

Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit. Für n achsenparallele Labels gleicher Größe berechnet es eine -Approximation.

Abschätzung aktive Gesamthöhe

Lemma.

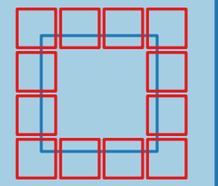
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.

- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.

Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz.

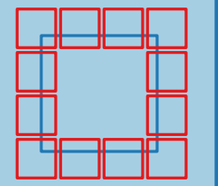
Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit. Für n achsenparallele Labels gleicher Größe berechnet es eine (1/8)-Approximation.

Abschätzung aktive Gesamthöhe

Lemma.
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.
- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

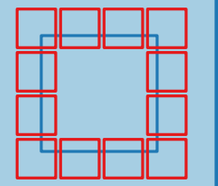
Satz.
Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit. Für n achsenparallele Labels gleicher Größe berechnet es eine (1/8)-Approximation

Abschätzung aktive Gesamthöhe

Lemma.
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.
- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz. Ebenen werden mit Faktor $(1 - 4\varepsilon)$ statt 2 kleiner

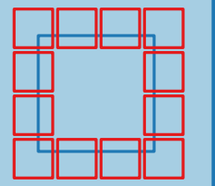
Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit. Für n achsenparallele Labels gleicher Größe berechnet es eine (1/8)-Approximation

Abschätzung aktive Gesamthöhe

Lemma.
Sei $\mathcal{S}$ optimale Lösung und $\mathcal{A}$ Lösung von LayeredGreedy. Wenn keinem Label in $\mathcal{A}$ mehr als c Labels in $\mathcal{S}$ zugewiesen werden, dann ist $\mathcal{A}$ eine $\frac{1}{2c}$-Approximation.

Beweisidee.
- Labels aus $\mathcal{A}$ können durch maximal c Labels aus $\mathcal{S}$ ersetzt werden.
- Labels können fast $2\times$ so lang aktiv sein (bis kurz vor nächster Ebene)

Lemma.
Sei L ein aktives Label in Ebene π_i. Sei ℓ die Seitenlänge der Spur von L in π_i. Sei L' ein inaktives Label, das L zugewiesen wurde. Dann hat die Spur von L' Seitenlänge $\ell' \geq \ell/3$ und schneidet den Rand von L.



➡ $c \leq 12$

Satz. Ebenen werden mit Faktor $(1-4\varepsilon)$ statt 2 kleiner

Für n achsenparallele Quadrate berechnet LayeredGreedy eine (1/24)-Approximation in $\mathcal{O}(n \log^3 n)$ Zeit. Für n achsenparallele Labels gleicher Größe berechnet es eine (1/8)-Approximation / eine $(1/4-\varepsilon)$-Approximation in $\mathcal{O}(n \log n \log(n/\varepsilon)/\varepsilon)$ Zeit.