

Vorlesung Algorithmische Kartografie

Punktbeschriftung in Dynamischen Karten

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

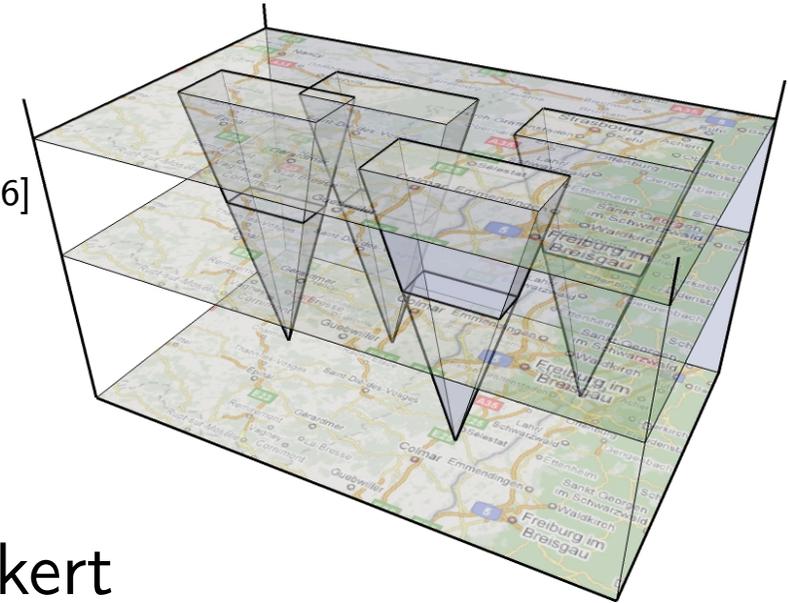
Benjamin Niedermann · **Martin Nöllenburg**
02.06.2015



Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

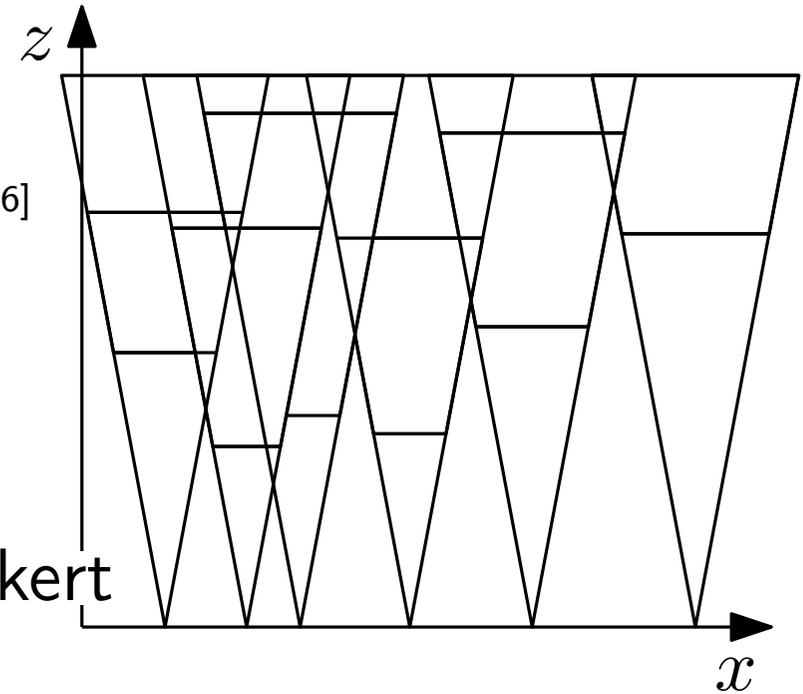
- (inverser) Maßstab auf z -Achse
- horizontaler Schnitt bei $z = z_0$
liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
⇒ **kein Springen**



Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

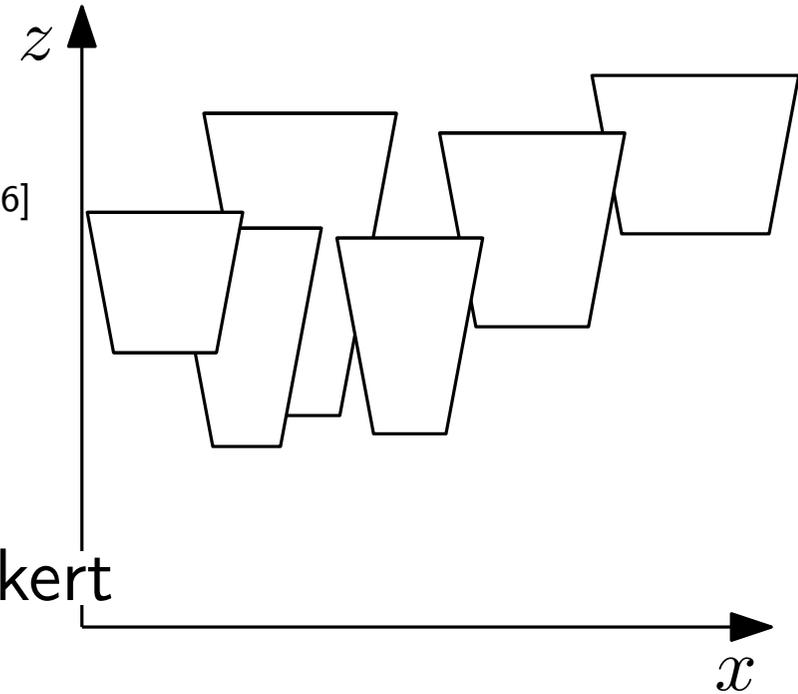
- (inverser) Maßstab auf z -Achse
- horizontaler Schnitt bei $z = z_0$
liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
⇒ **kein Springen**
- verfügbares Maßstabsintervall S_L für jedes L



Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

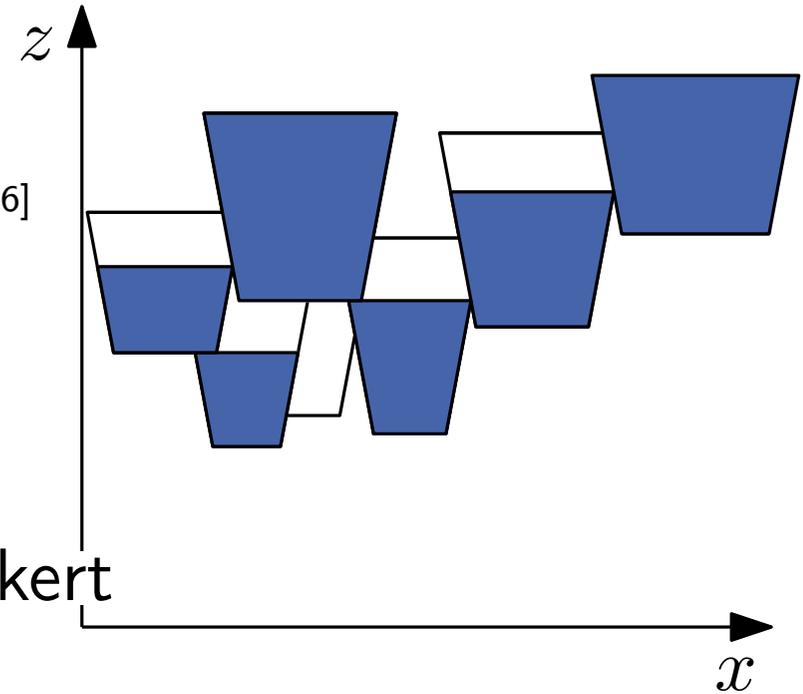
- (inverser) Maßstab auf z -Achse
- horizontaler Schnitt bei $z = z_0$
liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
⇒ **kein Springen**
- verfügbares Maßstabsintervall S_L für jedes L



Dynamisches Beschriftungsmodell

[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**
- verfügbares Maßstabsintervall S_L für jedes L
- berechne ein **aktives Intervall** $A_L \subseteq S_L$
⇒ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (bzw. allg. Labelkörper) induzieren ⇒ **gültige Beschriftung**
- **Ziel:** maximiere **aktive Gesamthöhe** $\sum_L |A_L|$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

$E \leftarrow Q.\text{extractMax}$

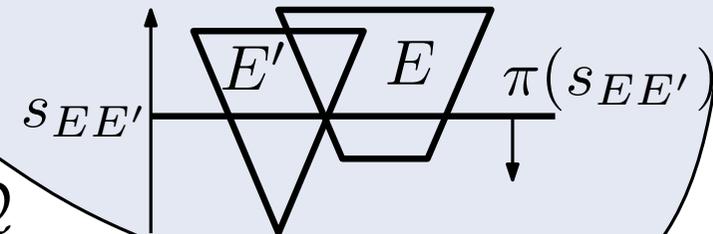
foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q

$s_{EE'}$: größter z -Wert, so dass
 $E \cap E' \cap \pi(s_{EE'}) = \emptyset$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

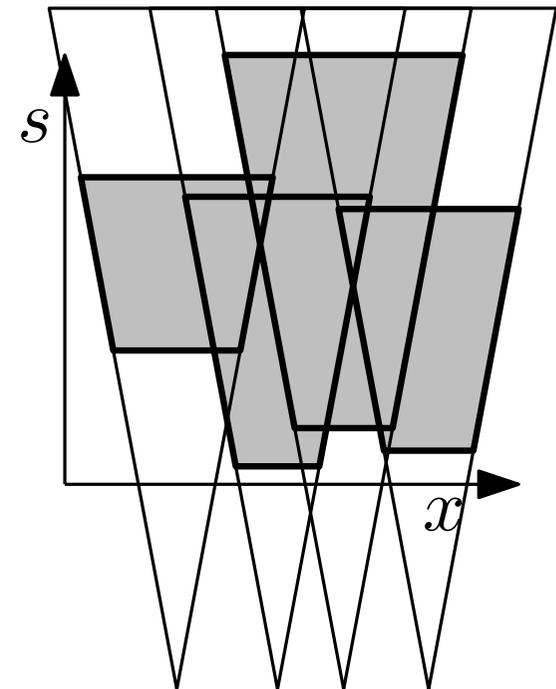
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

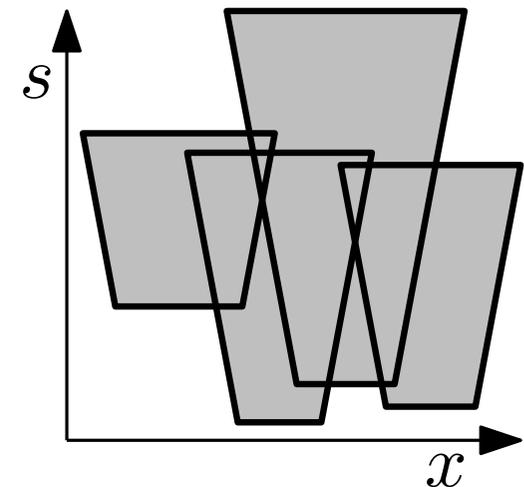
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

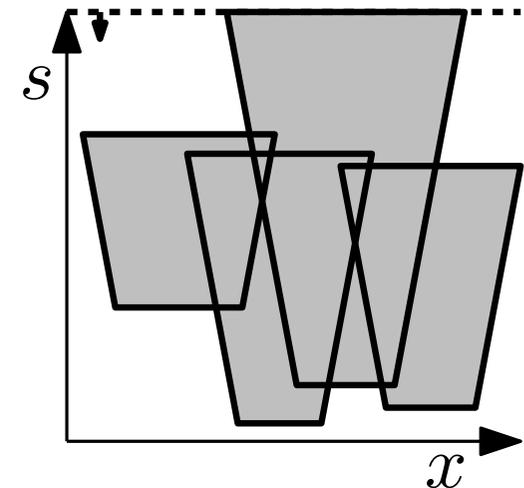
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

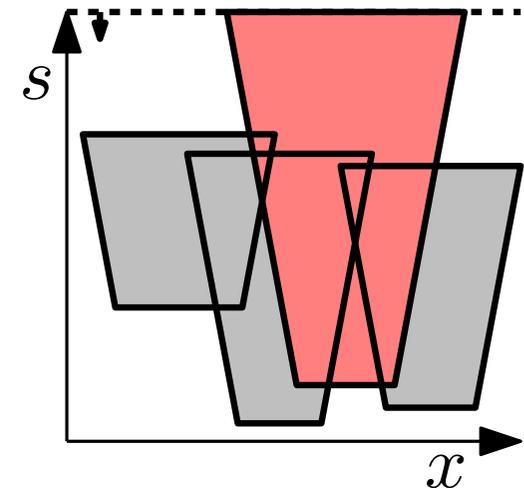
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

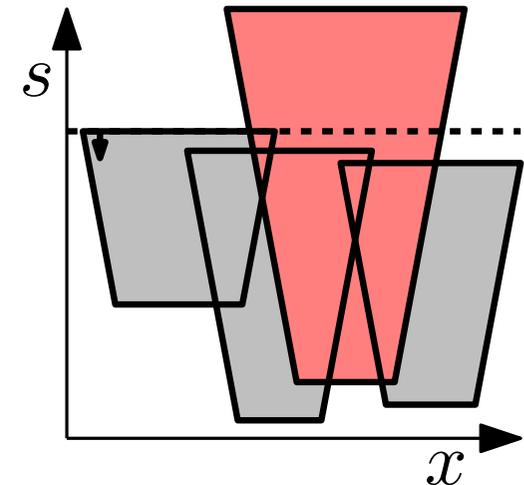
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

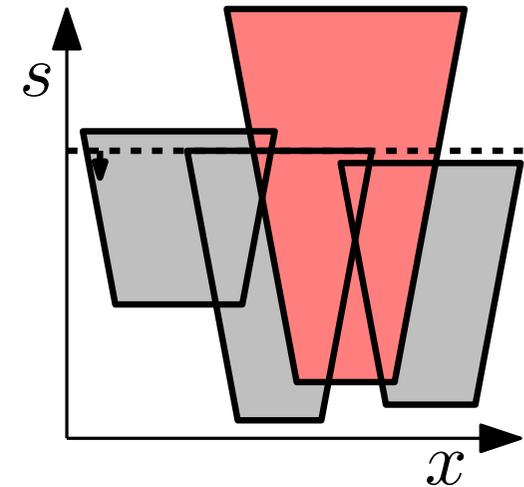
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

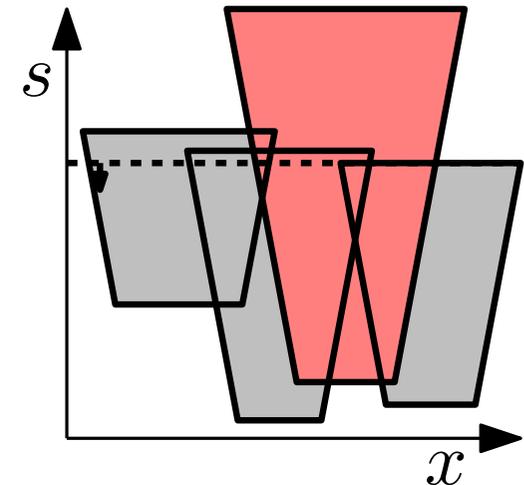
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

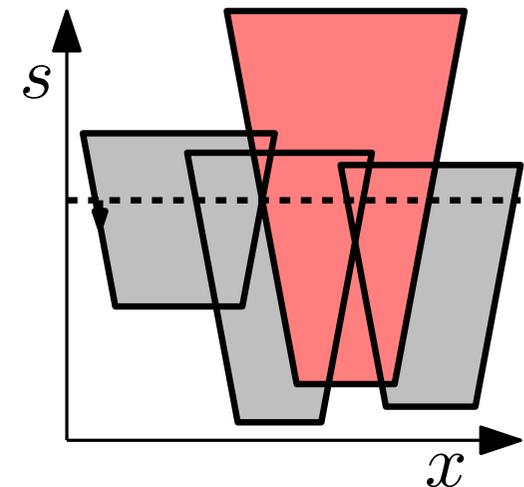
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

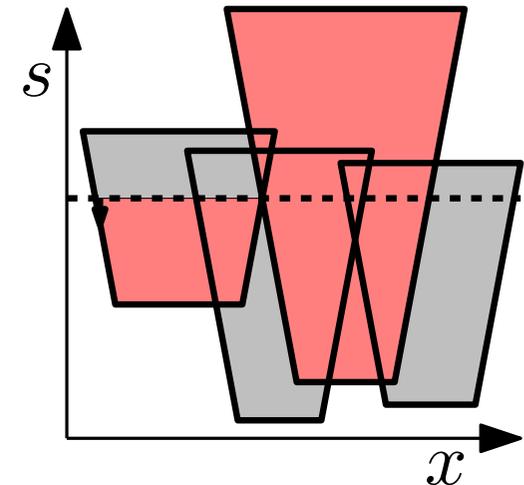
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

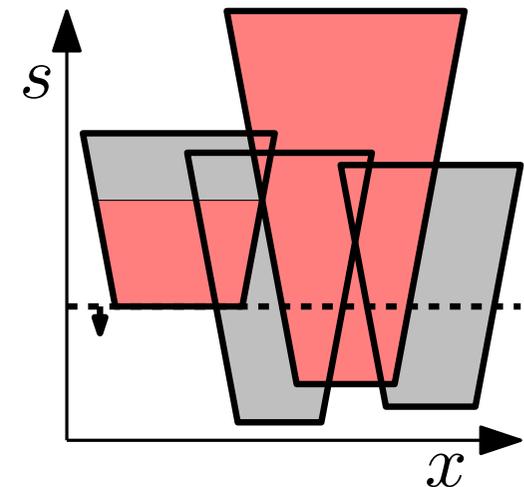
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

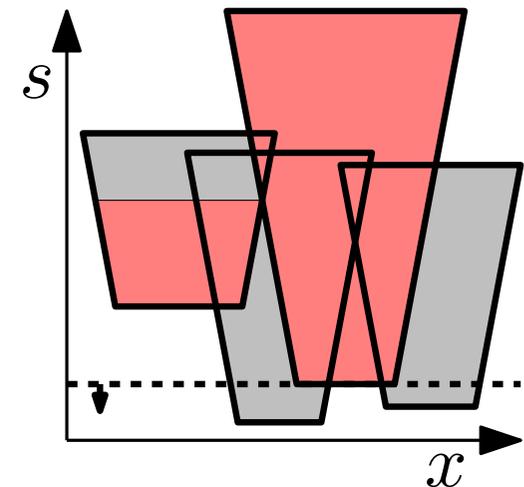
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

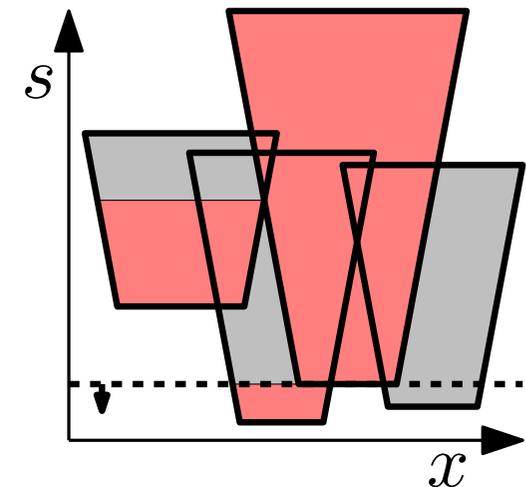
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

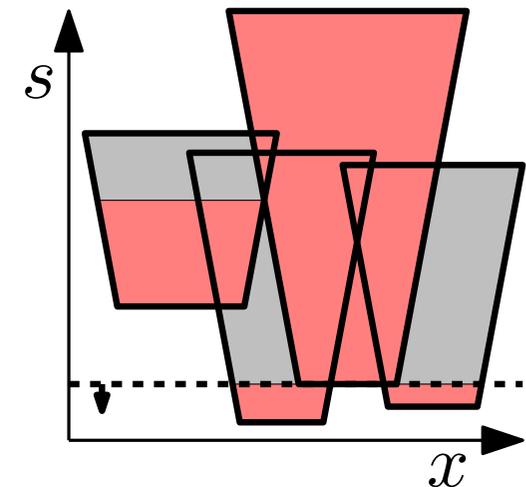
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

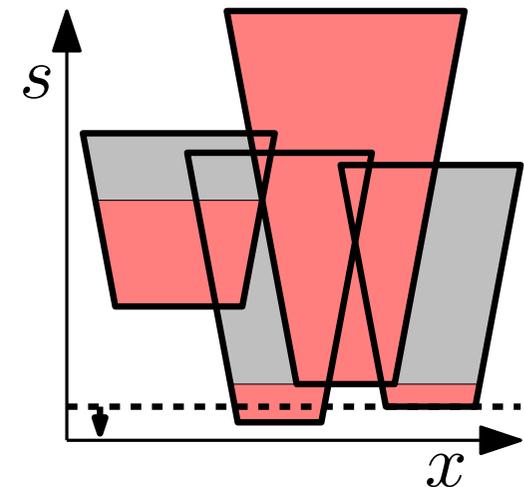
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

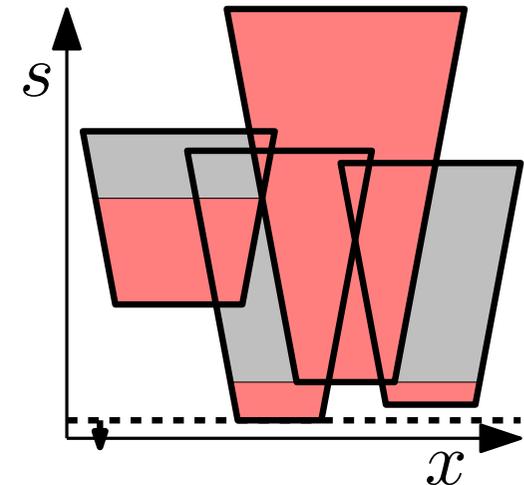
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

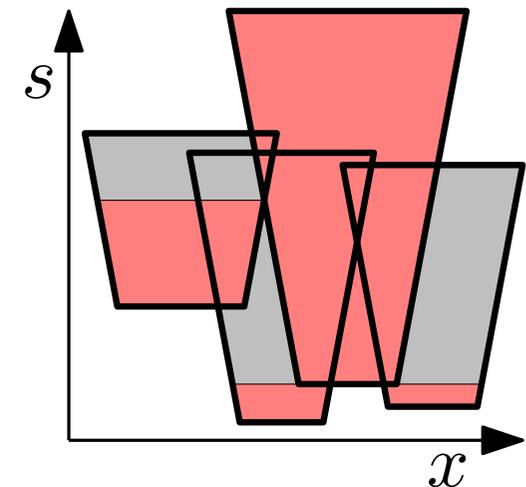
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



Lösung Algorithmus

Greedy Top-Down Sweep Algorithmus

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

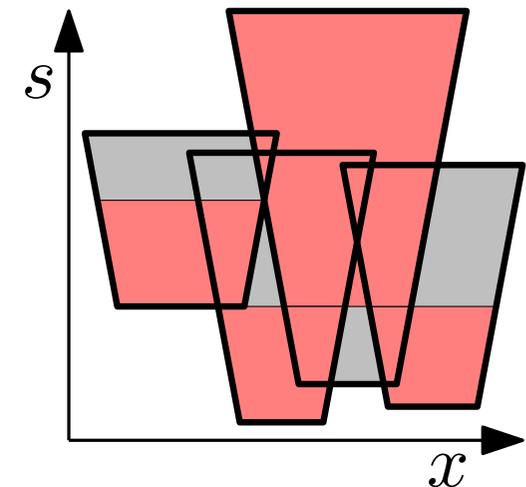
$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q



optimale Lösung

Algorithmus 1

Input: Menge \mathcal{E} von (offenen) Labelpyramiden, verfügbare Intervalle (s_E, S_E) für alle $E \in \mathcal{E}$

Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

$Q \leftarrow$ Priority Queue für \mathcal{E} lexikogr. absteigend sortiert nach (A_E, S_E)

while $Q \neq \emptyset$ **do**

$E \leftarrow Q.\text{extractMax}$

foreach $E' \in Q$ **do**

if $E' \cap E \neq \emptyset$ **then**

$A_{E'} \leftarrow \min\{A_{E'}, s_{EE'}\}$

if $A_{E'} = a_{E'}$ **then** lösche E' aus Q

Lemma 1: Algorithmus 1 berechnet eine gültige Beschriftung und lässt sich naiv in $O(n^2)$ Zeit und $O(n)$ Platz implementieren.

Lemma 2: Sei \mathcal{E} eine Menge von *Labelkörpern*. Falls gilt, dass jedes $E \in \mathcal{E}$ für jeden z -Wert s nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine $(1/c)$ -Approximation.

Lemma 2: Sei \mathcal{E} eine Menge von *Labelkörpern*. Falls gilt, dass jedes $E \in \mathcal{E}$ für jeden z -Wert s nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine $(1/c)$ -Approximation.

Begriffe:

- $E \in \mathcal{E}$ **blockiert** $E' \in \mathcal{E}$ am Wert s in einer Lösung \mathcal{A} , falls $a_E \leq s \leq A_E$ und $tr_s(E) \cap tr_s(E') \neq \emptyset$
- die **Spur** $tr_s(E)$ eines Labelkörpers $E \in \mathcal{E}$ zum Wert s ist der Schnitt von E mit der Ebene $\pi(s) : z = s$
- $E, E' \in \mathcal{E}$ sind **unabhängig** am Wert s , falls $tr_s(E) \cap tr_s(E') = \emptyset$

Lemma 2: Sei \mathcal{E} eine Menge von *Labelkörpern*. Falls gilt, dass jedes $E \in \mathcal{E}$ für jeden z -Wert s nicht mehr als c paarweise unabhängige Labelkörper blockiert, dann liefert Algorithmus 1 eine $(1/c)$ -Approximation.

Begriffe:

- $E \in \mathcal{E}$ **blockiert** $E' \in \mathcal{E}$ am Wert s in einer Lösung \mathcal{A} , falls $a_E \leq s \leq A_E$ und $tr_s(E) \cap tr_s(E') \neq \emptyset$
- die **Spur** $tr_s(E)$ eines Labelkörpers $E \in \mathcal{E}$ zum Wert s ist der Schnitt von E mit der Ebene $\pi(s) : z = s$
- $E, E' \in \mathcal{E}$ sind **unabhängig** am Wert s , falls $tr_s(E) \cap tr_s(E') = \emptyset$

Was ist c für den Fall von quadratischen Einheitslabeln?
Was ist c für den Fall einheitlich großer Rechtecke?

Approximation für Einheitslabel

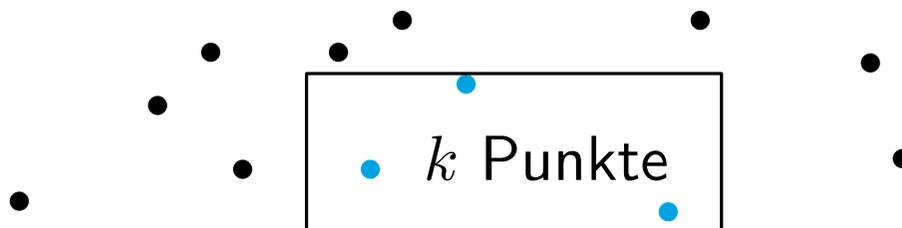
Satz 2: Für n achsenparallele Label gleicher Größe berechnet Algorithmus 1 eine $1/4$ -Approximation der optimalen aktiven Gesamthöhe. Der Algorithmus lässt sich in $O((n + k) \log^2 n)$ Zeit und $O(n \log n)$ Platz implementieren.

k ist die Anzahl von Seitenflächen-Schnitten der Labelpyramiden

Satz 2: Für n achsenparallele Label gleicher Größe berechnet Algorithmus 1 eine $1/4$ -Approximation der optimalen aktiven Gesamthöhe. Der Algorithmus lässt sich in $O((n + k) \log^2 n)$ Zeit und $O(n \log n)$ Platz implementieren.

k ist die Anzahl von Seitenflächen-Schnitten der Labelpyramiden

Range Tree: geometrische Datenstruktur zur Beantwortung rechteckiger Bereichsanfragen für eine Punktmenge P der Größe n .
Aufbau: $O(n \log n)$, Speicher: $O(n \log n)$, Query: $O(k + \log^2 n)$



Übungen

- 1) Betrachten Sie den Greedy Top-Down Sweep Algorithmus nun für kreisförmige Label einheitlicher Größe. Welchen Approximationsfaktor liefert das Blockadelemma?

Übungen

- 1) Betrachten Sie den Greedy Top-Down Sweep Algorithmus nun für kreisförmige Label einheitlicher Größe. Welchen Approximationsfaktor liefert das Blockadelemma?
- 2) In der Beschreibung des Greedy Top-Down Sweep Algorithmus haben wir die Labelkörper E lexikographisch nach (A_E, S_E) sortiert betrachtet. Zeigen Sie, dass eine andere Reihenfolge zweier kongruenter, aber in z -Richtung verschobener Labeltrapeze E, E' mit $A_E = A_{E'}$ die Approximationsgarantie verschlechtern kann.

Algorithmus 2

Input: Menge \mathcal{E} von quadratischen Labelpyramiden, verfügbare Intervalle $(0, S_{\max})$ für alle $E \in \mathcal{E}$

Output: aktive Intervalle $(0, A_E)$ für alle $E \in \mathcal{E}$

foreach $E \in \mathcal{E}$ **do** initialisiere E als inaktiv; $A_E \leftarrow 0$

for $i = 0$ **to** $\log_2 n$ **do** // Phase i

$s_i \leftarrow S_{\max}/2^i$

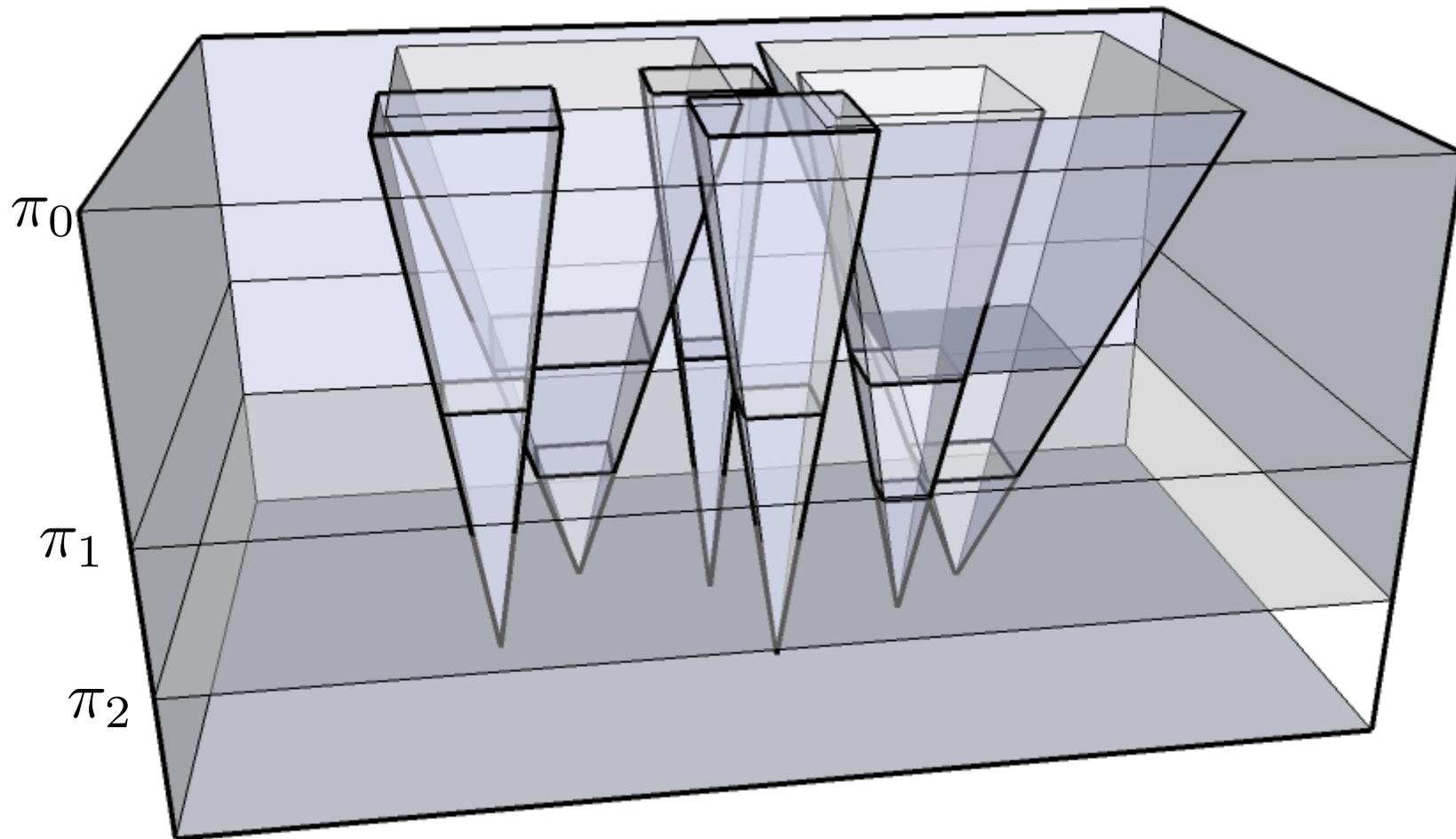
$\mathcal{C}_i \leftarrow$ Menge inaktiver Spuren in $\pi_i = \pi(s_i)$, die keine aktiven Spuren in π_i schneiden

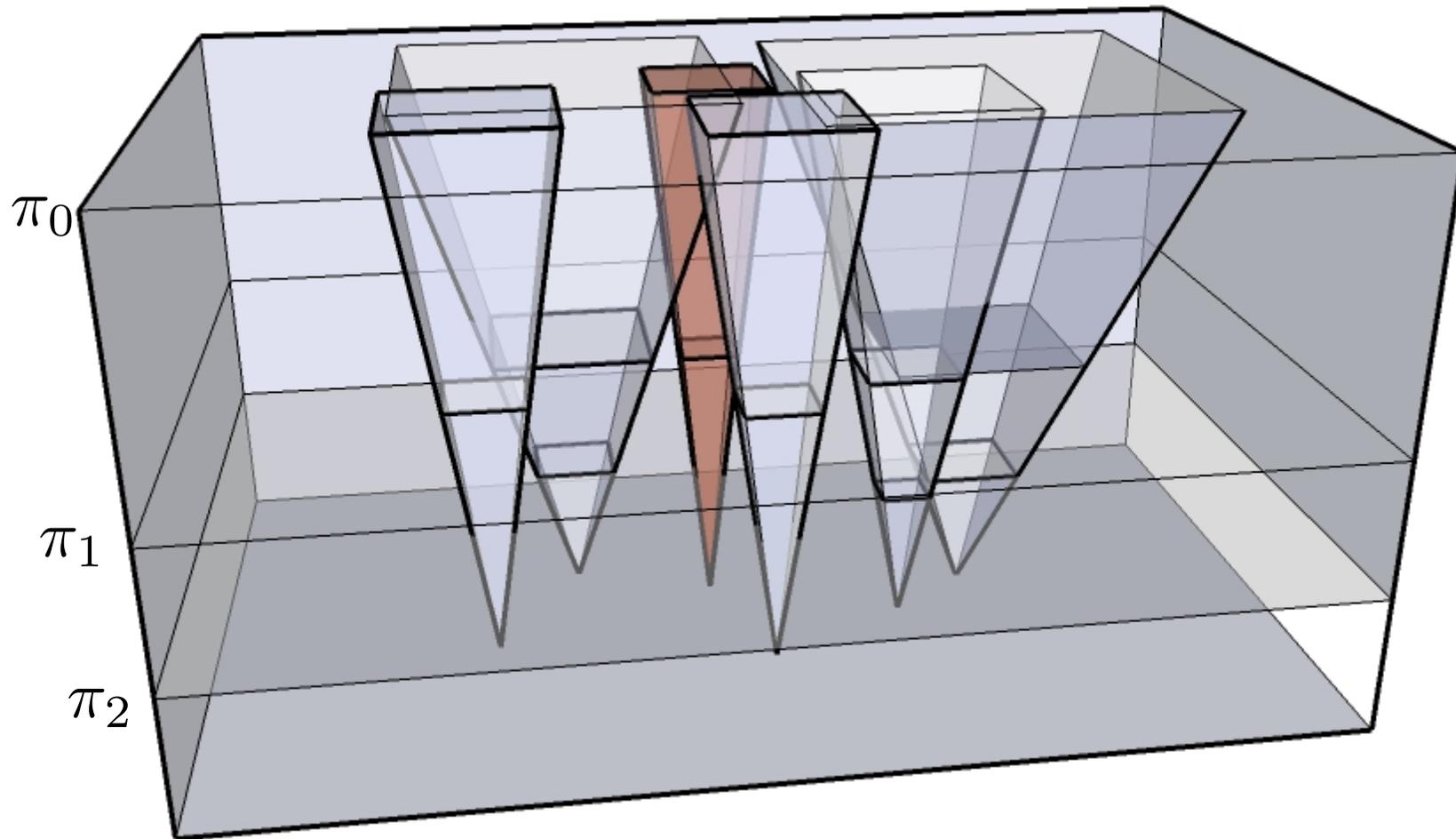
while $\mathcal{C}_i \neq \emptyset$ **do**

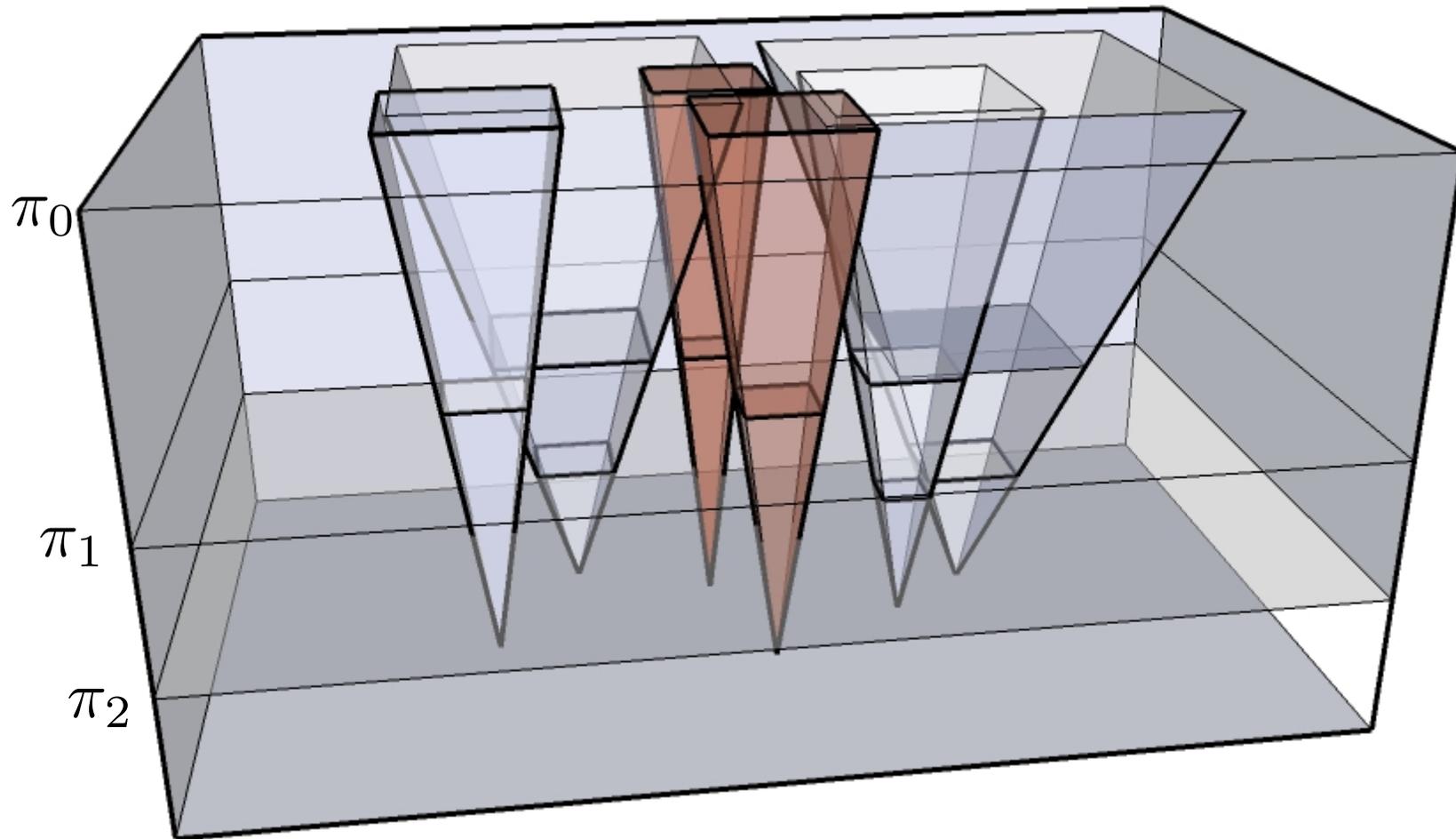
$T \leftarrow$ kleinste Spur in \mathcal{C}_i ; $E \leftarrow$ Pyramide zu T

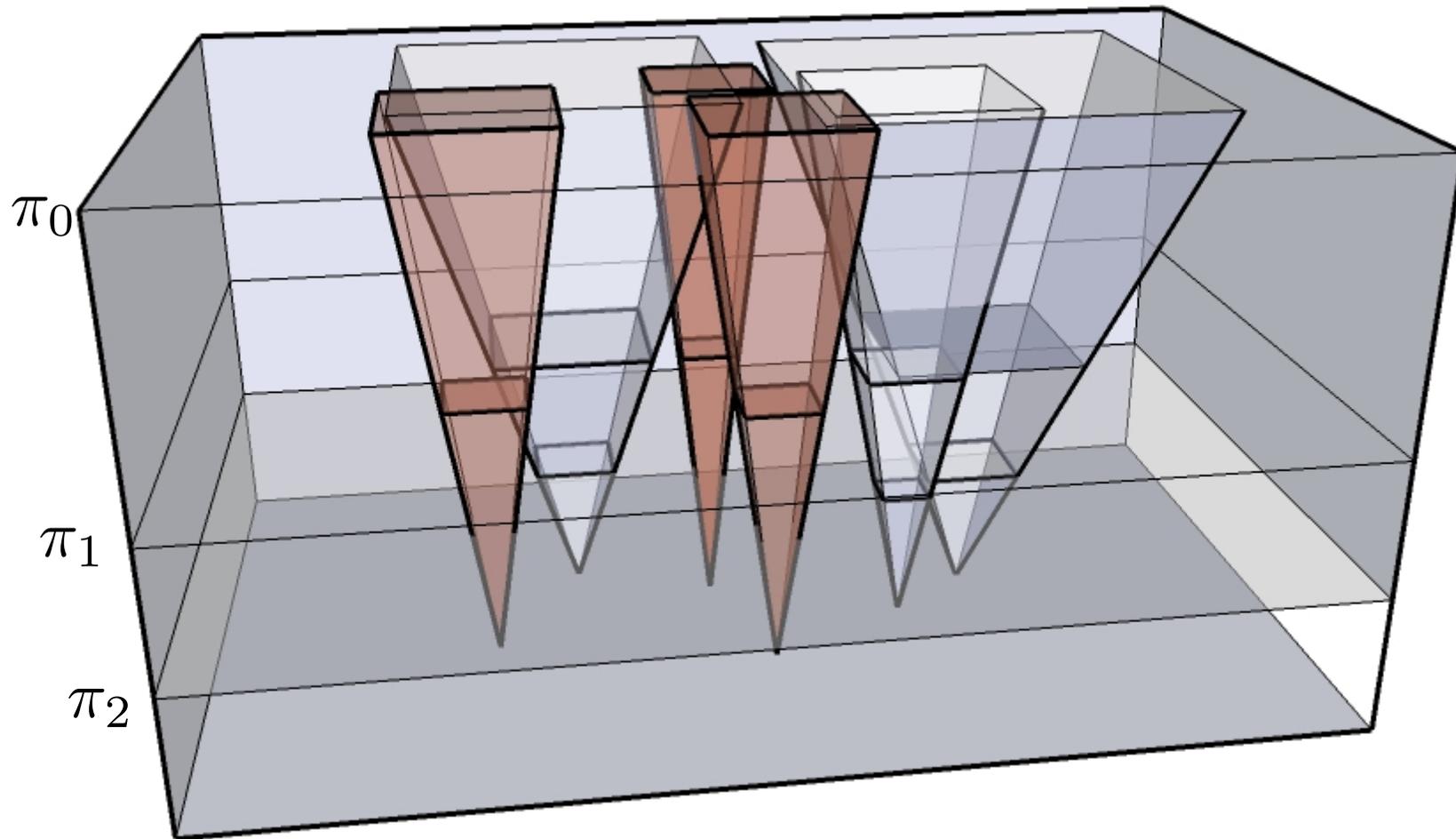
markiere E und T aktiv und setze $A_E \leftarrow s_i$

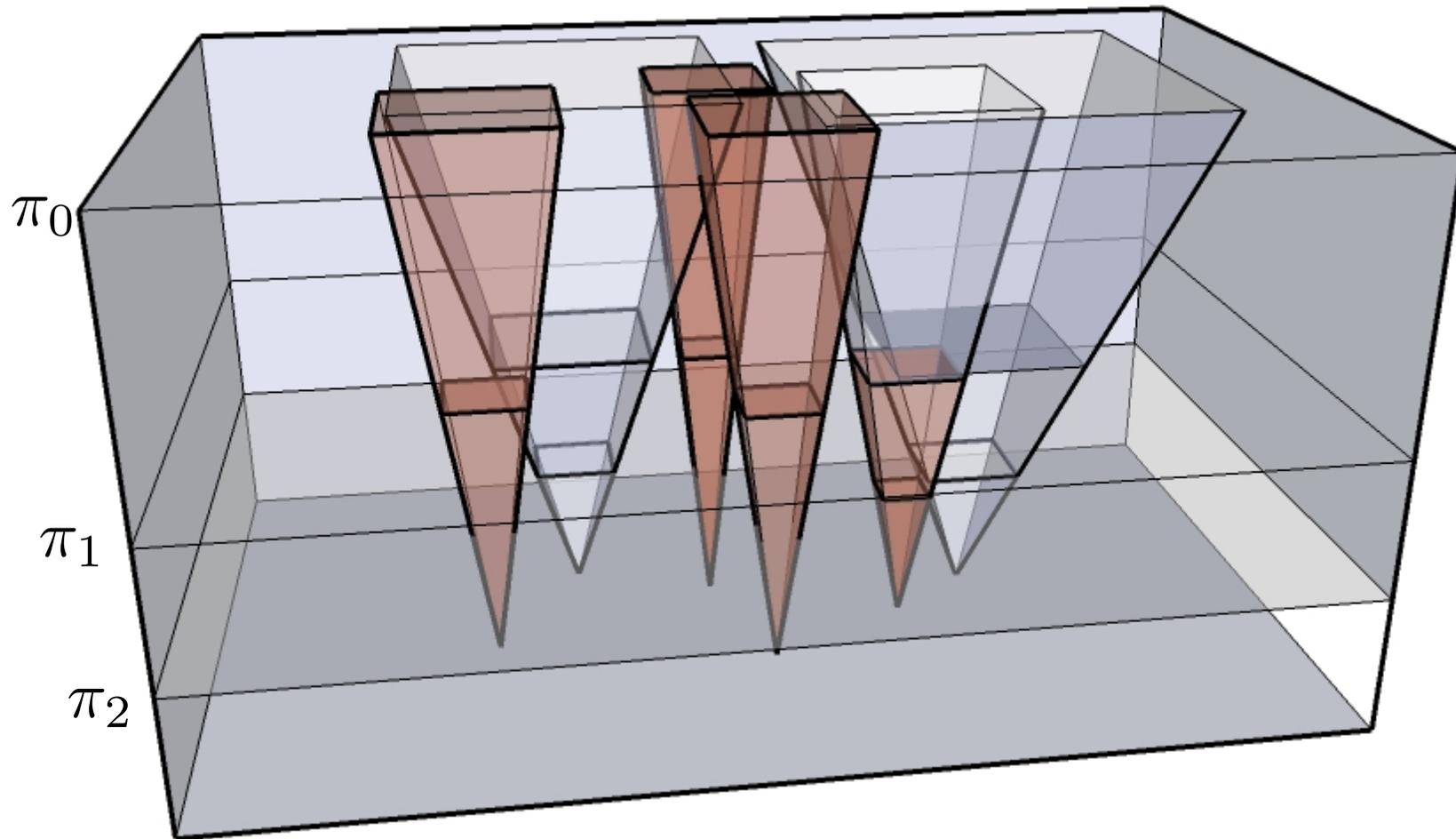
entferne T und alle geschnittenen Spuren aus \mathcal{C}_i

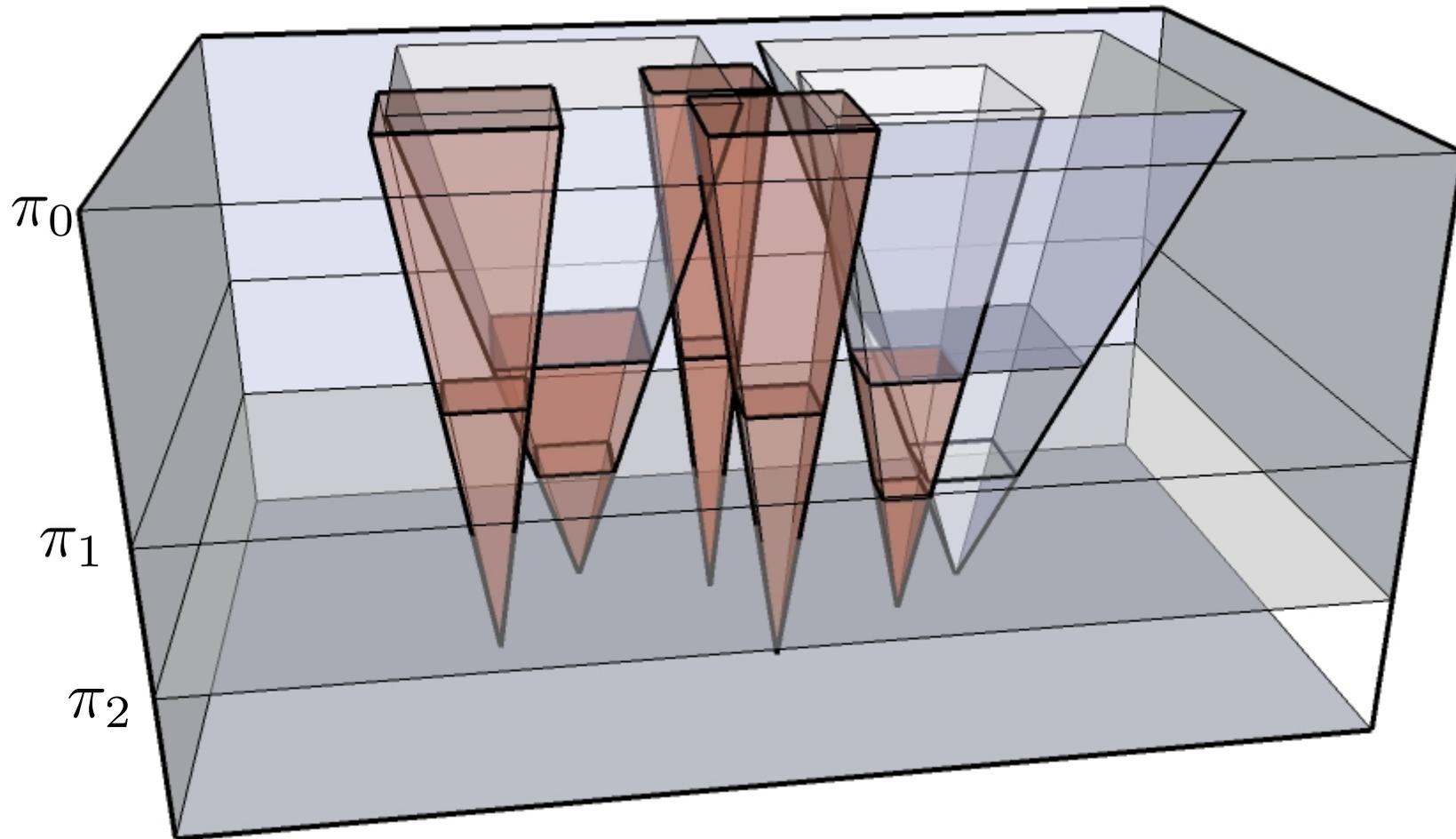


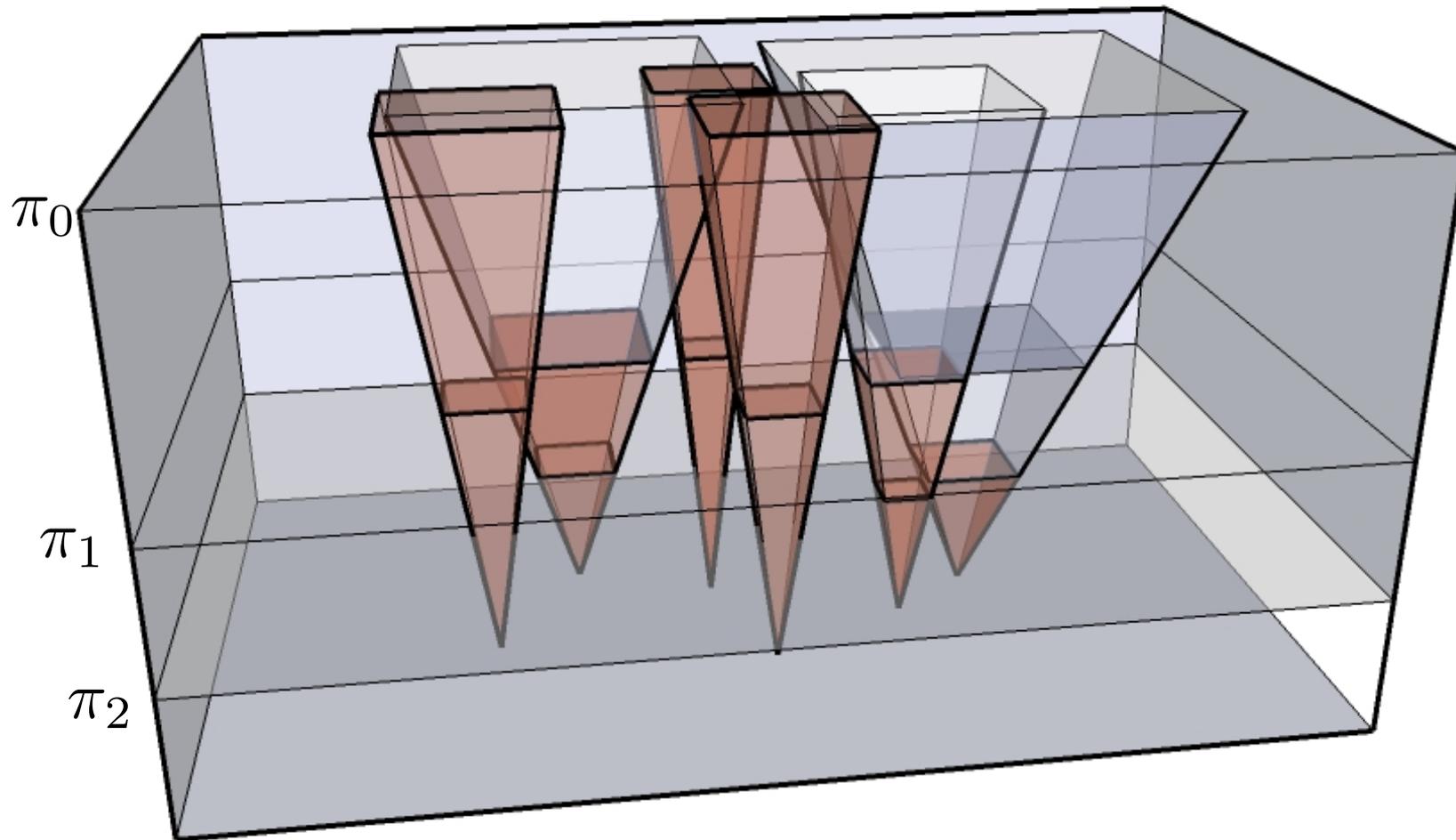












Blocker-Zuordnung

Am Ende von Phase i schneiden alle inaktiven Spuren eine aktive Spur in Ebene π_i , sie werden **blockiert**.

Sei T eine blockierte Spur. Wir weisen T einer aktiven Spur zu:

- (A) falls T anfangs in \mathcal{C}_i war, wurde T durch neu aktivierte Spur T' blockiert; weise T der Spur T' zu
- (B) sonst weise T einer beliebigen blockierenden Spur T' zu, die schon am Anfang von Phase i aktiv war

Blocker-Zuordnung

Am Ende von Phase i schneiden alle inaktiven Spuren eine aktive Spur in Ebene π_i , sie werden **blockiert**.

Sei T eine blockierte Spur. Wir weisen T einer aktiven Spur zu:

- (A) falls T anfangs in \mathcal{C}_i war, wurde T durch neu aktivierte Spur T' blockiert; weise T der Spur T' zu
- (B) sonst weise T einer beliebigen blockierenden Spur T' zu, die schon am Anfang von Phase i aktiv war

Lemma 3: Sei T eine aktive Spur in Ebene π_i mit Seitenlänge ℓ . Dann hat jede inaktive und T zugeordnete Spur Seitenlänge mind. $\ell/3$ und schneidet den Rand von T .

Blocker-Zuordnung

Am Ende von Phase i schneiden alle inaktiven Spuren eine aktive Spur in Ebene π_i , sie werden **blockiert**.

Sei T eine blockierte Spur. Wir weisen T einer aktiven Spur zu:

- (A) falls T anfangs in C_i war, wurde T durch neu aktivierte Spur T' blockiert; weise T der Spur T' zu
- (B) sonst weise T einer beliebigen blockierenden Spur T' zu, die schon am Anfang von Phase i aktiv war

Lemma 3: Sei T eine aktive Spur in Ebene π_i mit Seitenlänge ℓ . Dann hat jede inaktive und T zugeordnete Spur Seitenlänge mind. $\ell/3$ und schneidet den Rand von T .

Beweis: Sei $T = tr_i(E)$ aktiv und $T' = tr_i(E')$ inaktiv und T zugeordnet

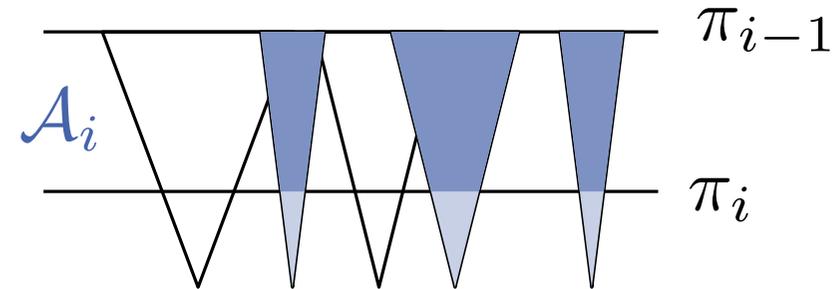
Fall (A) ist klar, Fall (B): Widerspruchsbeweis an der Tafel

Abschätzung aktive Gesamthöhe

Sei \mathcal{S} optimale Lösung und \mathcal{A} Lösung des Algorithmus 2.

Sei $\pi_{\log_2 n+1} = \pi(0)$.

Definiere \mathcal{S}_i und \mathcal{A}_i als Menge der aktiven Pyramidenstümpfe von \mathcal{S} bzw. \mathcal{A} zwischen π_{i-1} und π_i .

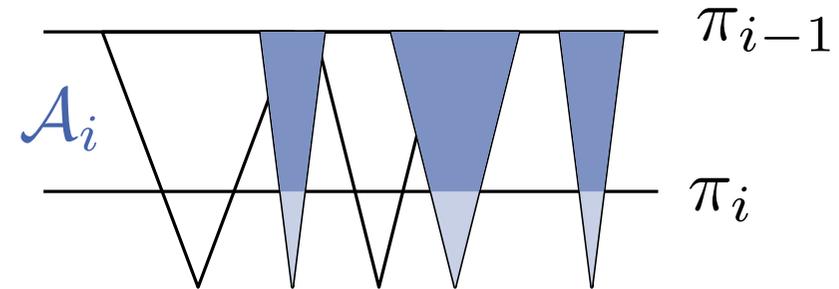


Abschätzung aktive Gesamthöhe

Sei \mathcal{S} optimale Lösung und \mathcal{A} Lösung des Algorithmus 2.

Sei $\pi_{\log_2 n+1} = \pi(0)$.

Definiere \mathcal{S}_i und \mathcal{A}_i als Menge der aktiven Pyramidenstümpfe von \mathcal{S} bzw. \mathcal{A} zwischen π_{i-1} und π_i .



Weise aktive Gesamthöhe $H(\mathcal{S}_i)$ der Gesamthöhe $H(\mathcal{A}_{i+1})$ zu;
weise $H(\mathcal{S}_{\log_2 n})$ und $H(\mathcal{S}_{\log_2 n+1})$ der Gesamthöhe $H(\mathcal{A}_1)$ zu.

Lemma 4: Es gilt $H(\mathcal{A}_1) \geq (H(\mathcal{S}_{\log_2 n}) + H(\mathcal{S}_{\log_2 n+1}))/4$.

Wenn nicht mehr als c Spuren in \mathcal{S} einer beliebigen Spur in \mathcal{A} zugeordnet sind, dann gilt für $1 \leq i < \log_2 n$: $H(\mathcal{A}_{i+1}) \geq H(\mathcal{S}_i)/(2c)$.

Satz 3: Für n achsenparallele quadratische Label beliebiger Größe berechnet Algorithmus 2 eine $1/24$ -Approximation der optimalen aktiven Gesamthöhe. Der Algorithmus lässt sich in $O(n \log^3 n)$ Zeit und $O(n \log n)$ Platz implementieren.

Satz 3: Für n achsenparallele quadratische Label beliebiger Größe berechnet Algorithmus 2 eine $1/24$ -Approximation der optimalen aktiven Gesamthöhe. Der Algorithmus lässt sich in $O(n \log^3 n)$ Zeit und $O(n \log n)$ Platz implementieren.

Abschlussfrage:

Was macht Algorithmus 2 für Label einheitlicher Größe?

Satz 3: Für n achsenparallele quadratische Label beliebiger Größe berechnet Algorithmus 2 eine $1/24$ -Approximation der optimalen aktiven Gesamthöhe. Der Algorithmus lässt sich in $O(n \log^3 n)$ Zeit und $O(n \log n)$ Platz implementieren.

Abschlussfrage:

Was macht Algorithmus 2 für Label einheitlicher Größe?

Ausblick: dynamisches Rotieren

