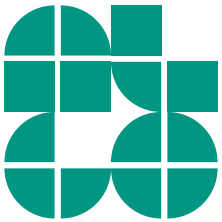
Übung Algorithmische Kartografie

Übungsblatt 6

LEHRSTUHL FÜR ALGORITHMIK I · INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Benjamin Niedermann

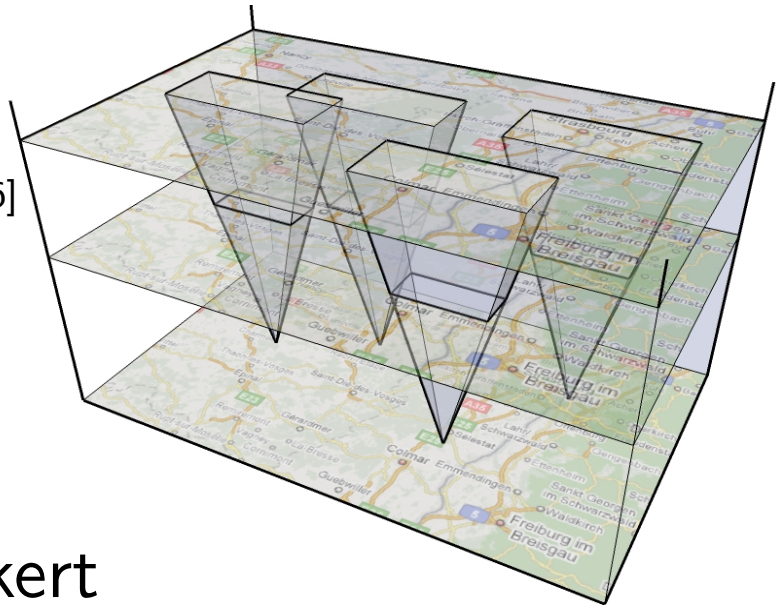
13.06.2013



3D Modell für dynamisches Beschriften

Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]



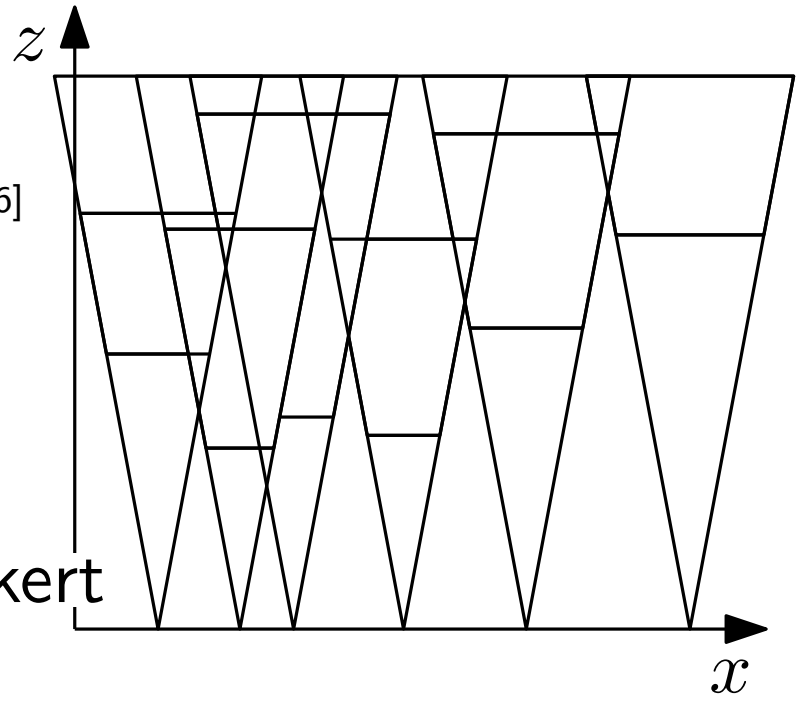
- (inverser) Maßstab auf z-Achse
- horizontale Scheibe bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert

  $\Rightarrow$ **kein Springen**

3D Modell für dynamisches Beschriften

Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

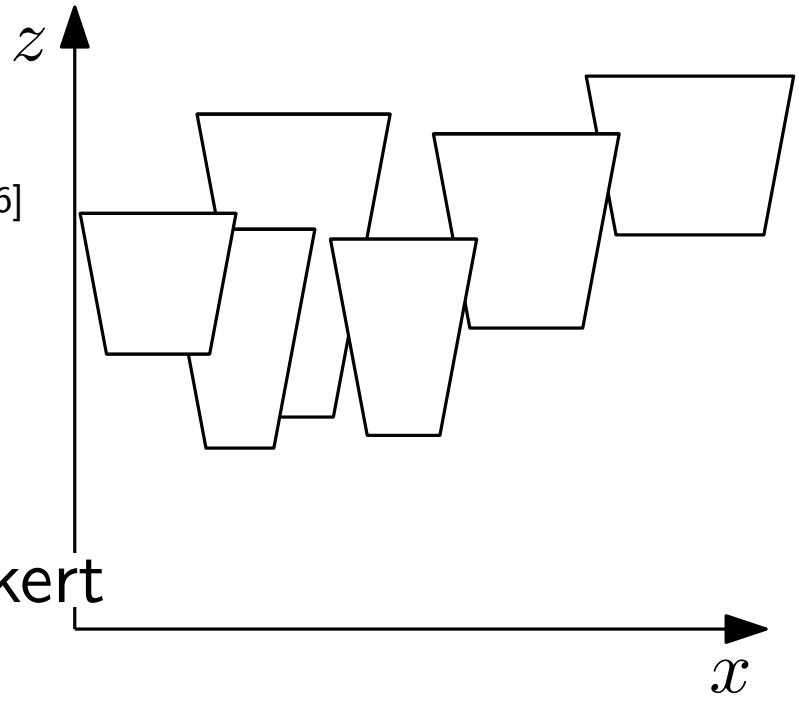


- (inverser) Maßstab auf z-Achse
- horizontale Scheibe bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
  $\Rightarrow$ **kein Springen**
- verfügbares Maßstabsintervall S_L für jedes L

3D Modell für dynamisches Beschriften

Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]

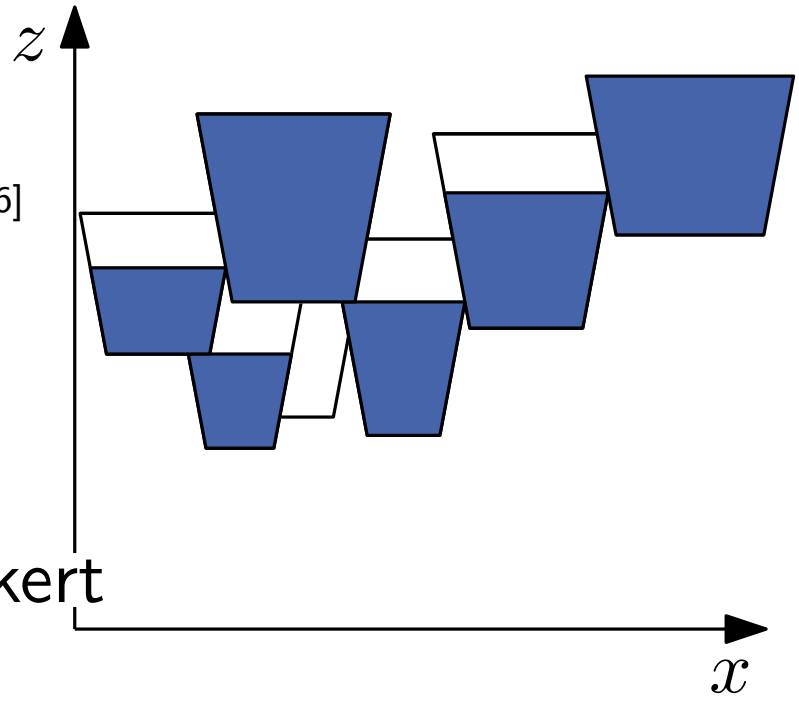


- (inverser) Maßstab auf z-Achse
- horizontale Scheibe bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
  $\Rightarrow$ **kein Springen**
- verfügbares Maßstabsintervall S_L für jedes L

3D Modell für dynamisches Beschriften

Dynamisches Beschriftungsmodell

[Been, Daiches, Yap '06]



- (inverser) Maßstab auf z-Achse
- horizontale Scheibe bei $z = z_0$ liefert Karte in Maßstab $1/z_0$
- jedes Label an festem Punkt verankert
  $\Rightarrow$ **kein Springen**
- verfügbares Maßstabsintervall S_L für jedes L
- berechne ein **aktives Intervall** $A_L \subseteq S_L$
  $\Rightarrow$ **kein Flackern**
- aktive Intervalle müssen disjunkte Pyramidenstümpfe (bzw. allg. Labelkörper) induzieren $\Rightarrow$ **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 $E \in \mathcal{E}$

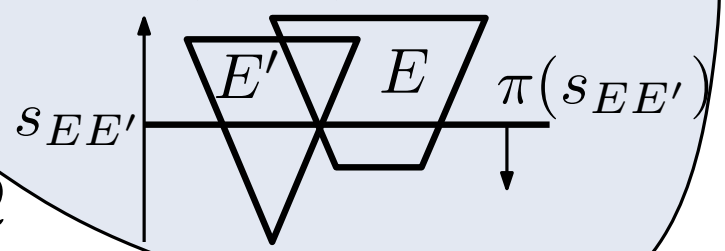
foreach $E \in \mathcal{E}$ **do** $(a_E, A_E) \leftarrow (s_E, S_E)$

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$

$s_{EE'}$: 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 $E \in \mathcal{E}$

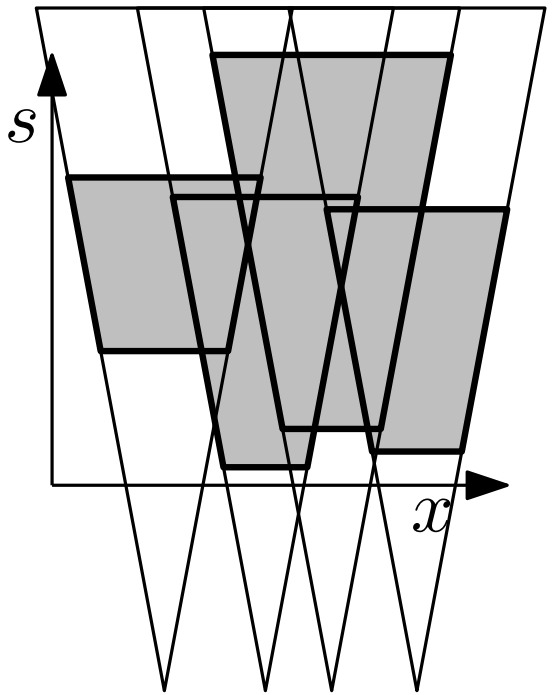
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

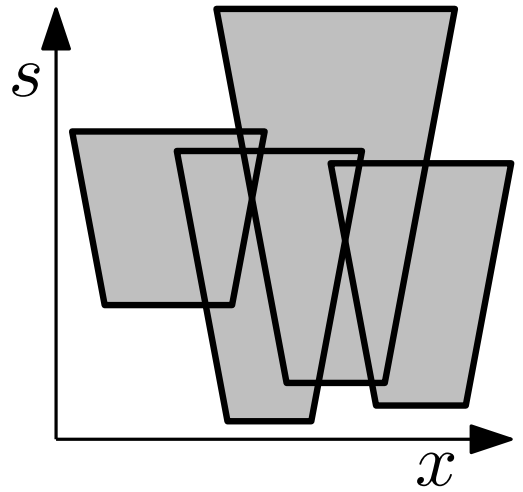
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

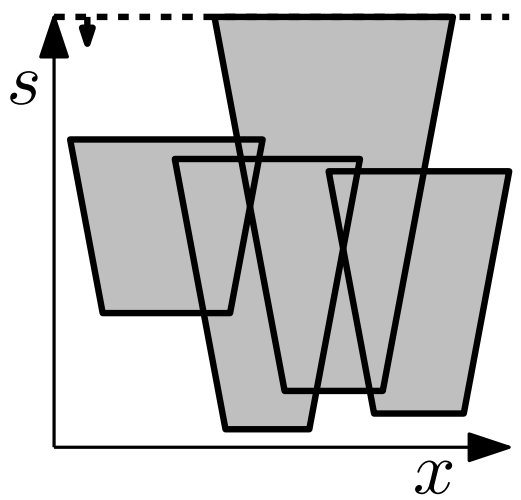
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

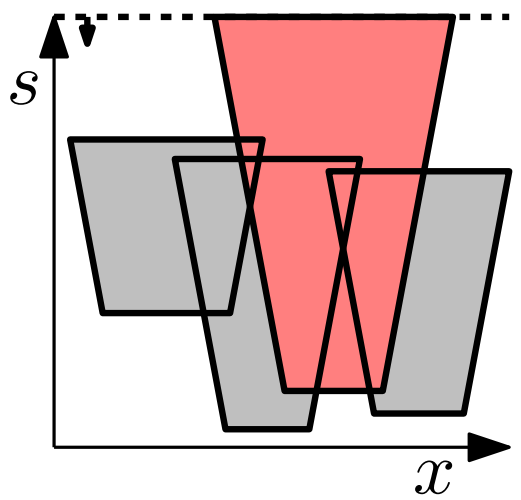
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

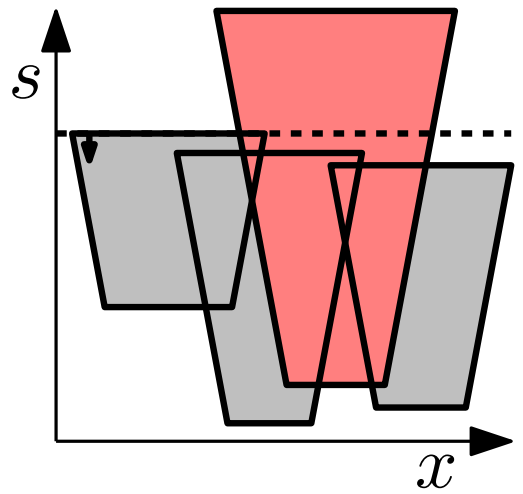
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

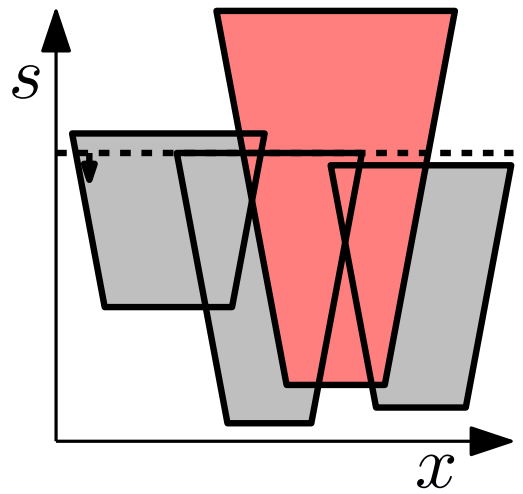
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

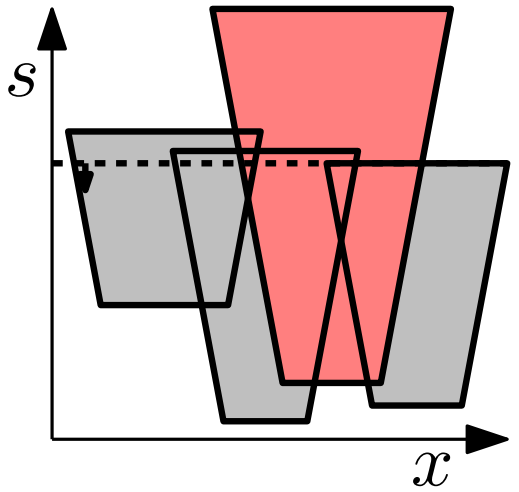
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

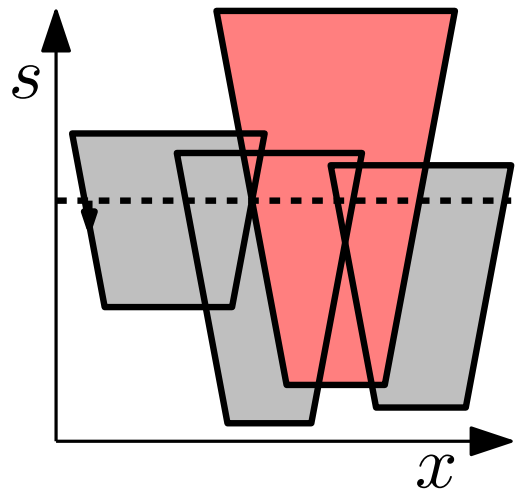
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

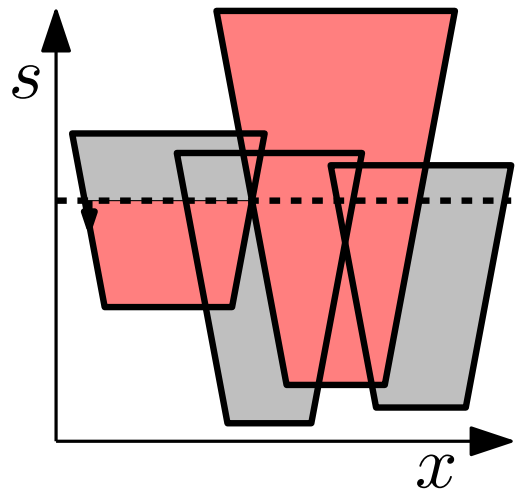
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

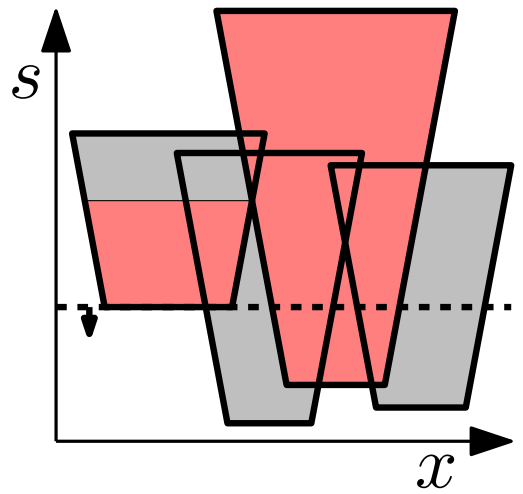
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

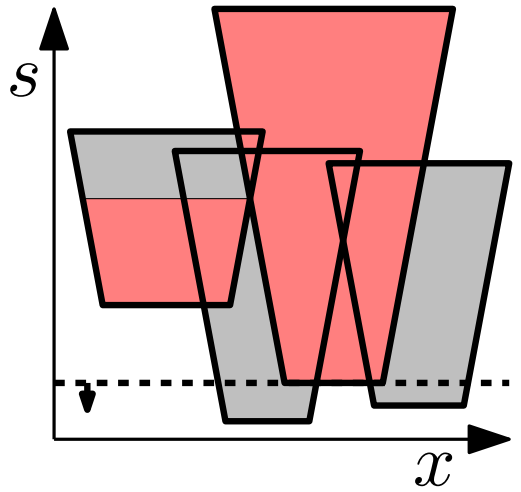
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

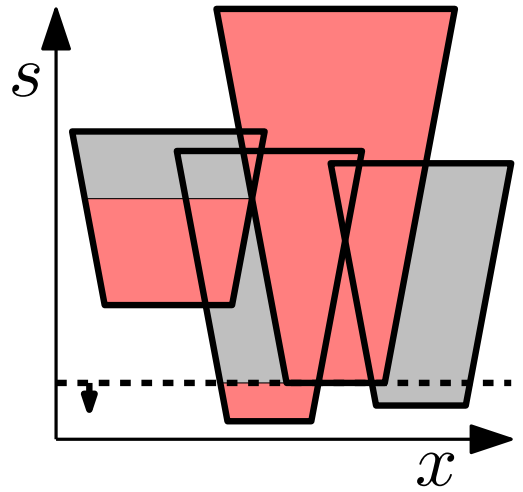
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

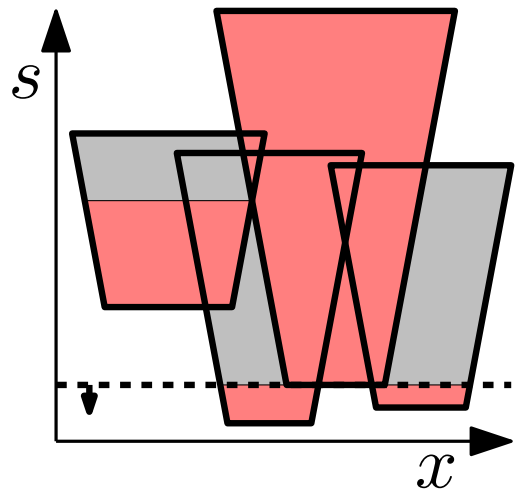
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

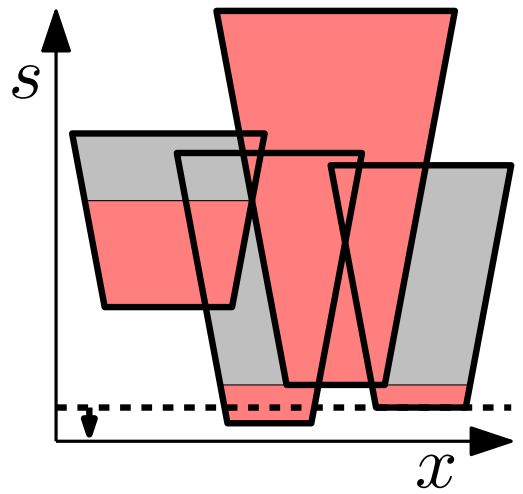
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

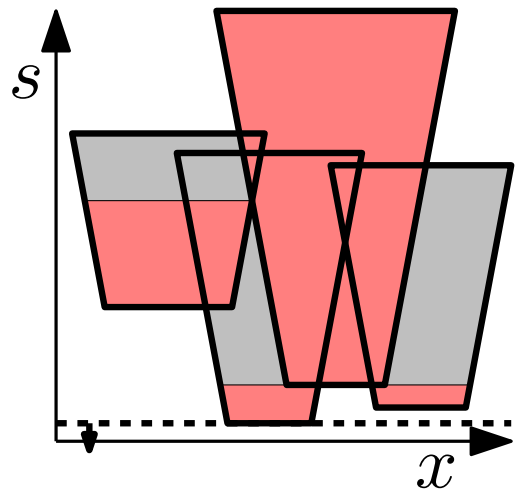
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

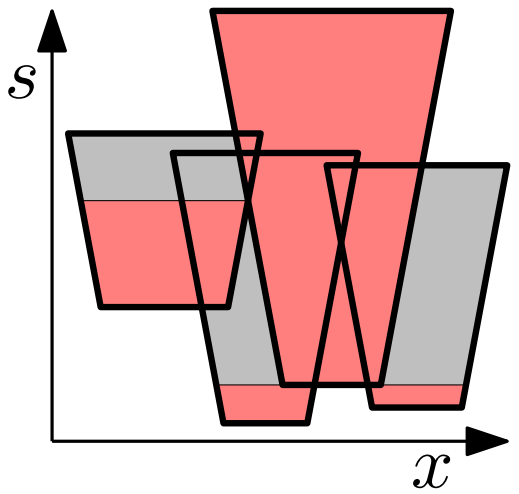
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



Lösung Algorithmus

Greedy Top-Down Sweep Algorithmus

Algorithmus 1

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

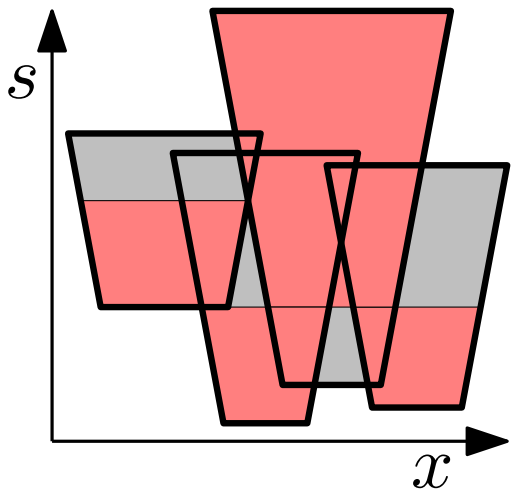
Output: aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$

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

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

while $\mathcal{Q} \neq \emptyset$ **do**

- $E \leftarrow \mathcal{Q}$.extractMax
- **foreach** $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$



optimale Lösung

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** $(a_E, A_E) \leftarrow (s_E, S_E)$

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

while $\mathcal{Q} \neq \emptyset$ **do**

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

  foreach $E' \in \mathcal{Q}$ **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 $\mathcal{Q}$

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

Blockade-Lemma

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.

Blockade-Lemma

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 $tr_s(E) \cap tr_s(E') = \emptyset$

Aufgabe 2

Gegeben: Menge $\mathcal{E}$ von Labelpyramiden mit verfügbaren Intervallen (s_E, S_E) für alle $E \in \mathcal{E}$.
Gesucht: Aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} (A_E - a_e)$ maximiert wird.

Aufgabe 2

Gegeben: Menge $\mathcal{E}$ von Labelpyramiden mit verfügbaren Intervallen (s_E, S_E) für alle $E \in \mathcal{E}$.
Gesucht: Aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} (A_E - a_e)$ maximiert wird.

Annahme:

1. Labels sind Quadrate.
2. Für jeden Skalierungsfaktor s und jedes $E \in \mathcal{E}$ gilt:

$$\frac{|tr_s(E)|}{|tr_s(E')|} \leq W,$$

wobei $|tr_s(E)|$ die Fläche der Spur $tr_s(E)$ bezeichnet.

Aufgabe 2

Gegeben: Menge $\mathcal{E}$ von Labelpyramiden mit verfügbaren Intervallen (s_E, S_E) für alle $E \in \mathcal{E}$.
Gesucht: Aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} (A_E - a_e)$ maximiert wird.

Annahme:

1. Labels sind Quadrate.
2. Für jeden Skalierungsfaktor s und jedes $E \in \mathcal{E}$ gilt:

$$\frac{|tr_s(E)|}{|tr_s(E')|} \leq W,$$

wobei $|tr_s(E)|$ die Fläche der Spur $tr_s(E)$ bezeichnet.

Welcher Approximationsfaktor ergibt sich?

Aufgabe 2

Gegeben: Menge $\mathcal{E}$ von Labelpyramiden mit verfügbaren Intervallen (s_E, S_E) für alle $E \in \mathcal{E}$.
Gesucht: Aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} (A_E - a_e)$ maximiert wird.

Annahme: Labels sind Kreise gleicher Größe.

Aufgabe 2

Gegeben: Menge $\mathcal{E}$ von Labelpyramiden mit verfügbaren Intervallen (s_E, S_E) für alle $E \in \mathcal{E}$.
Gesucht: Aktive Intervalle (a_E, A_E) für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} (A_E - a_e)$ maximiert wird.

Annahme: Labels sind Kreise gleicher Größe.

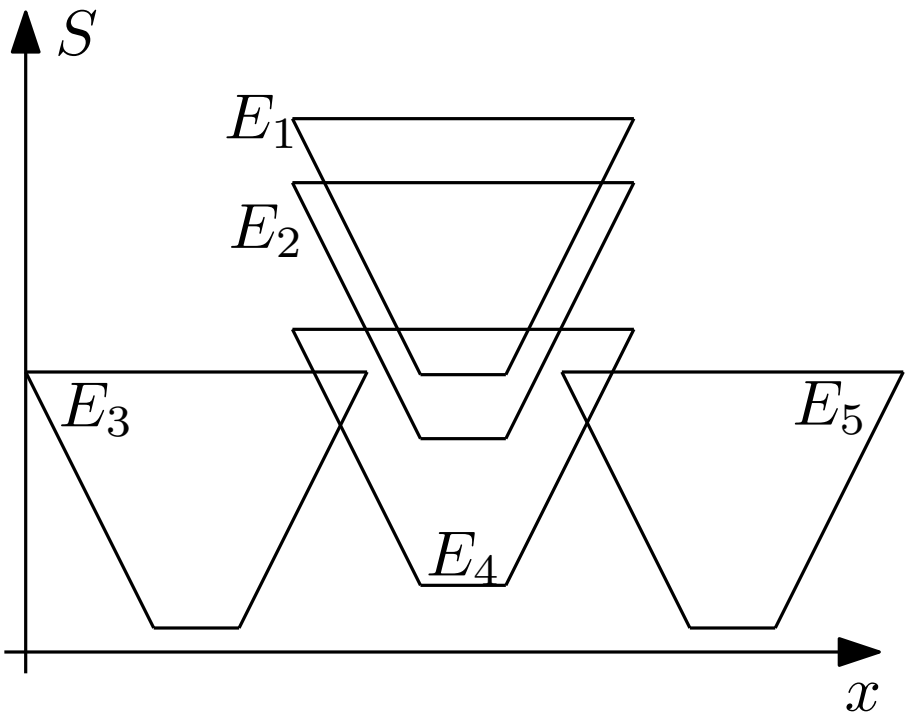
Welcher Approximationsfaktor ergibt sich?

Aufgabe 3

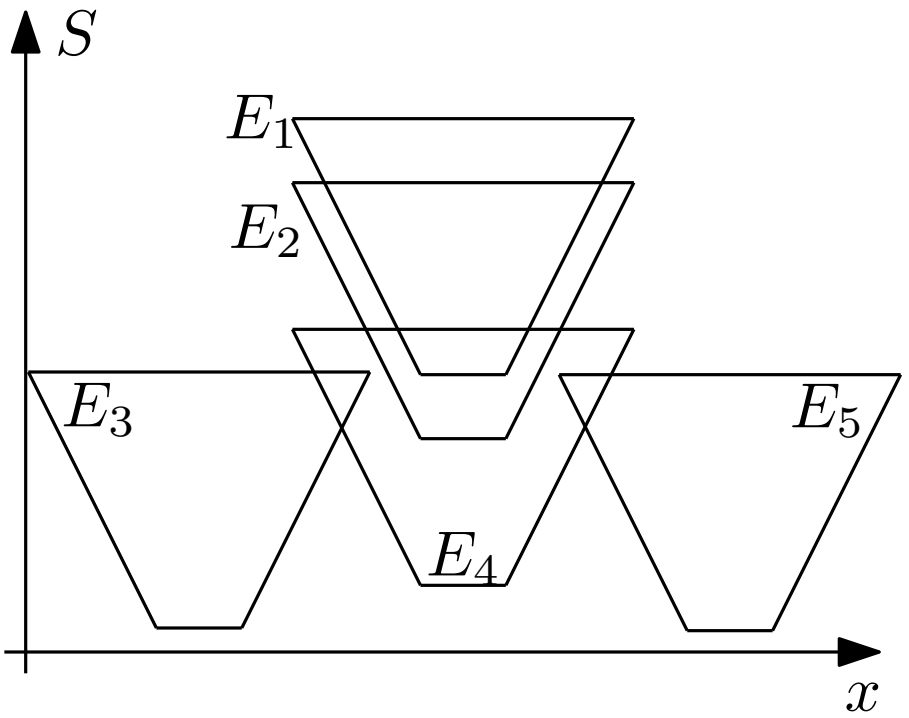
Was, wenn lexikografische Ordnung nicht verwendet wird?

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



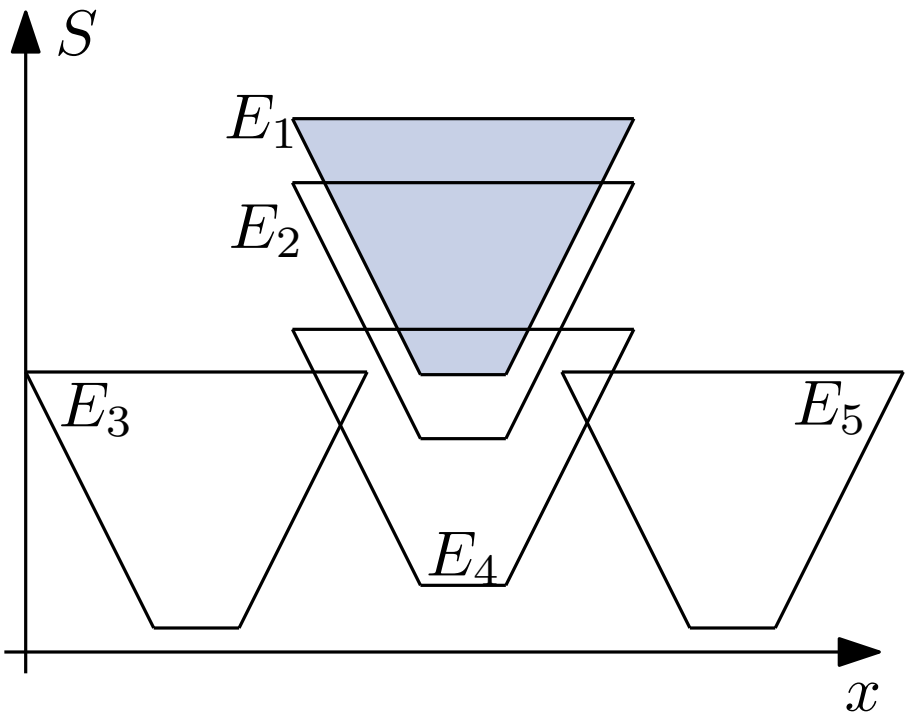
mit lexikografischer Ordnung



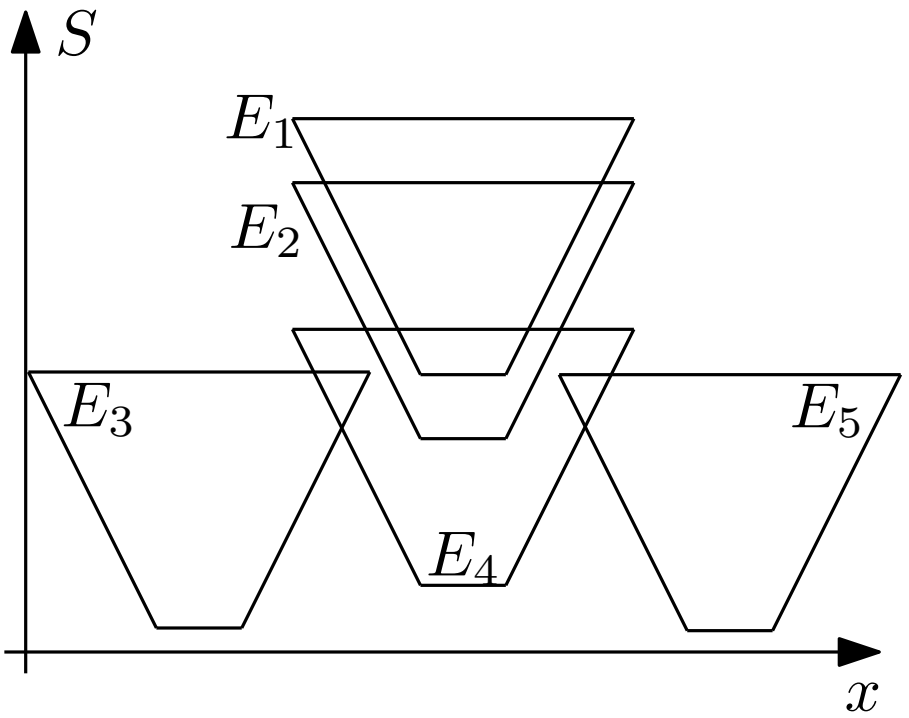
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



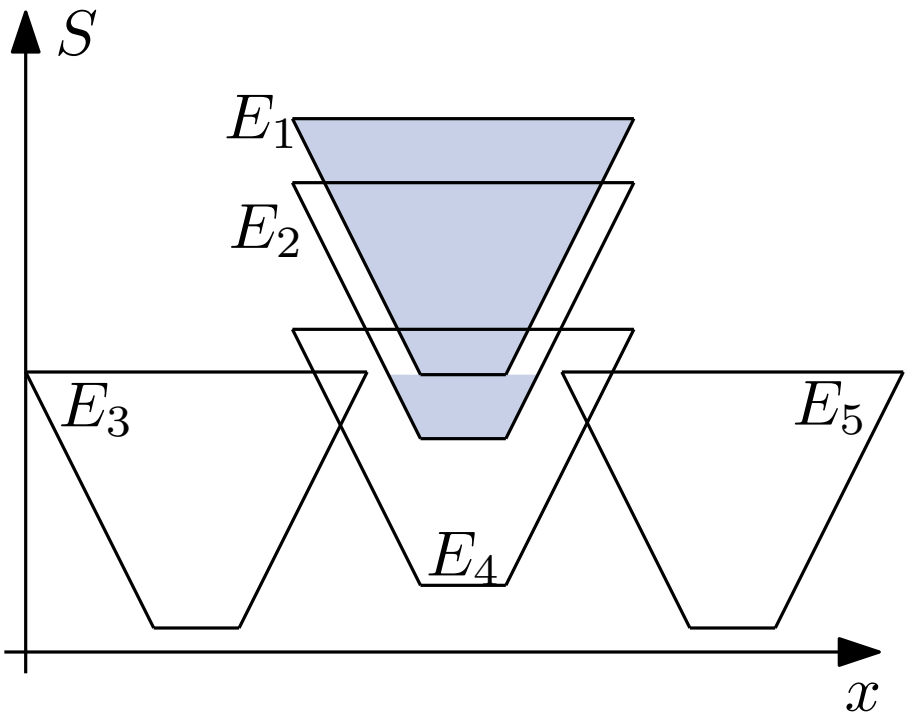
mit lexikografischer Ordnung



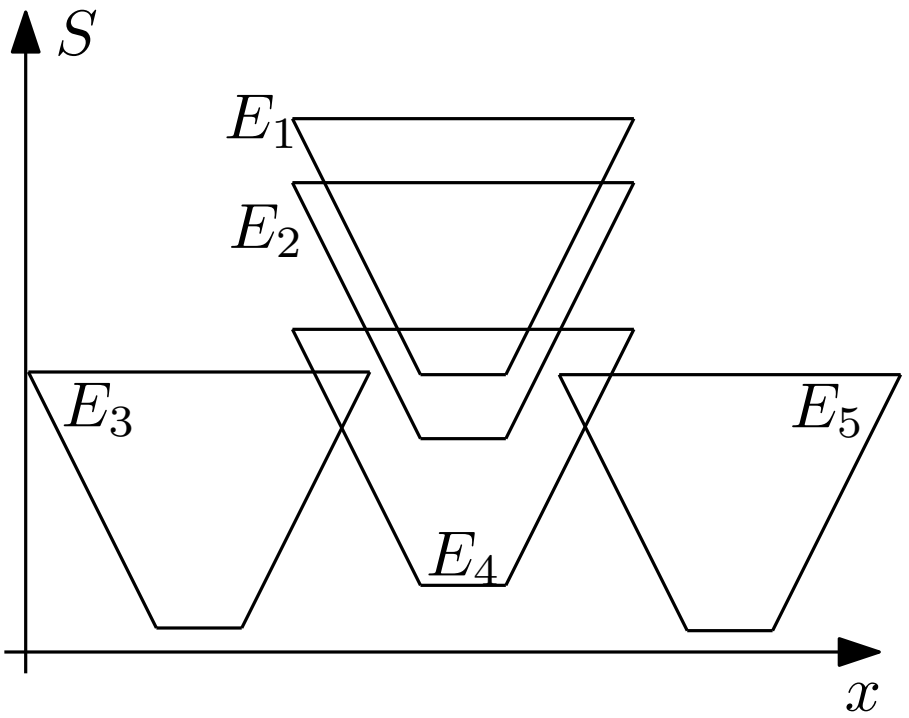
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



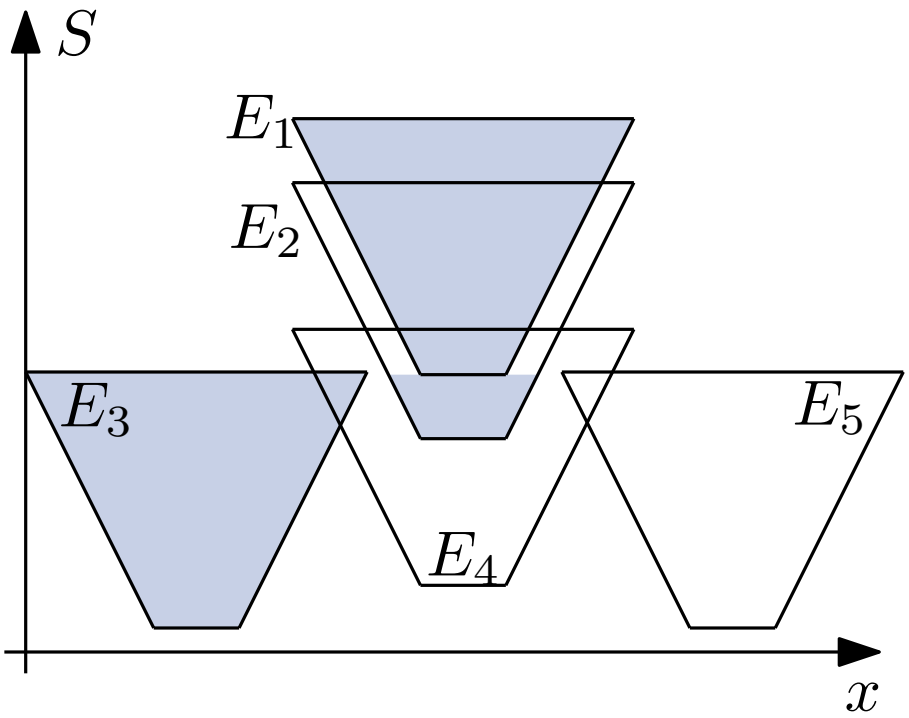
mit lexikografischer Ordnung



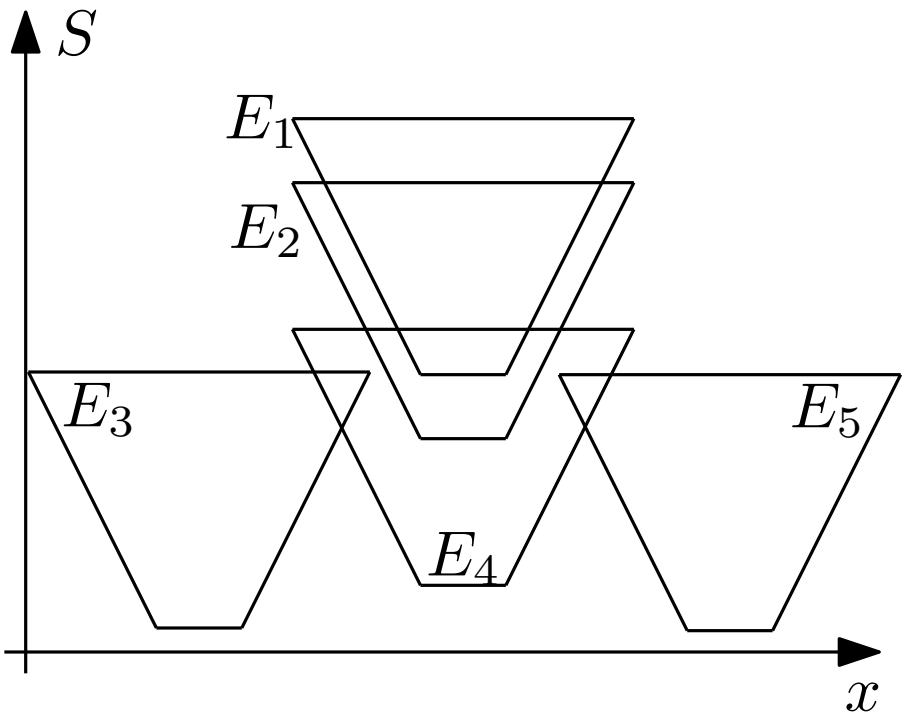
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



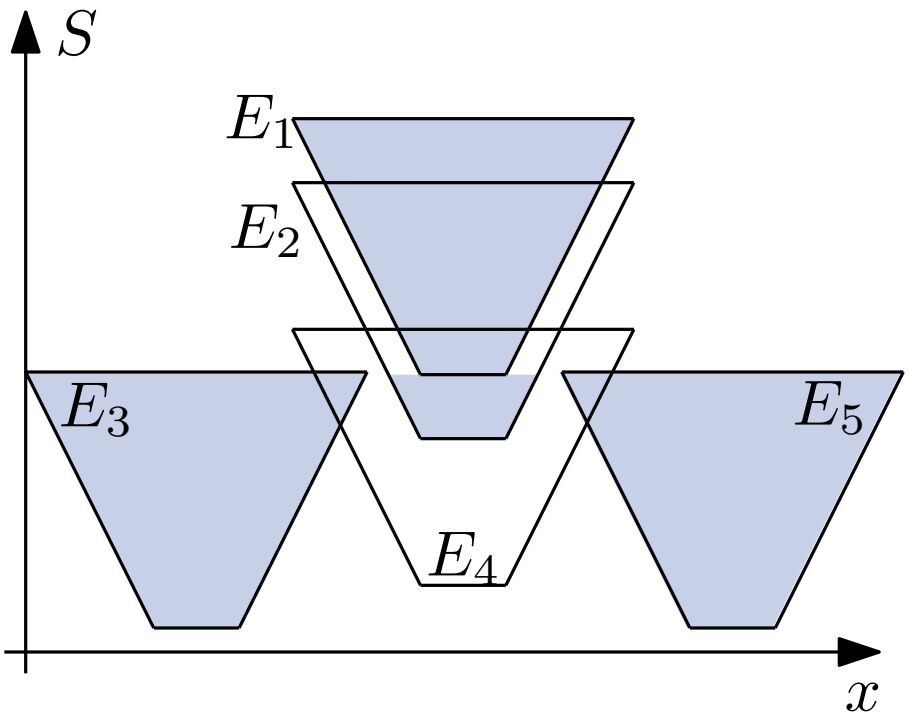
mit lexikografischer Ordnung



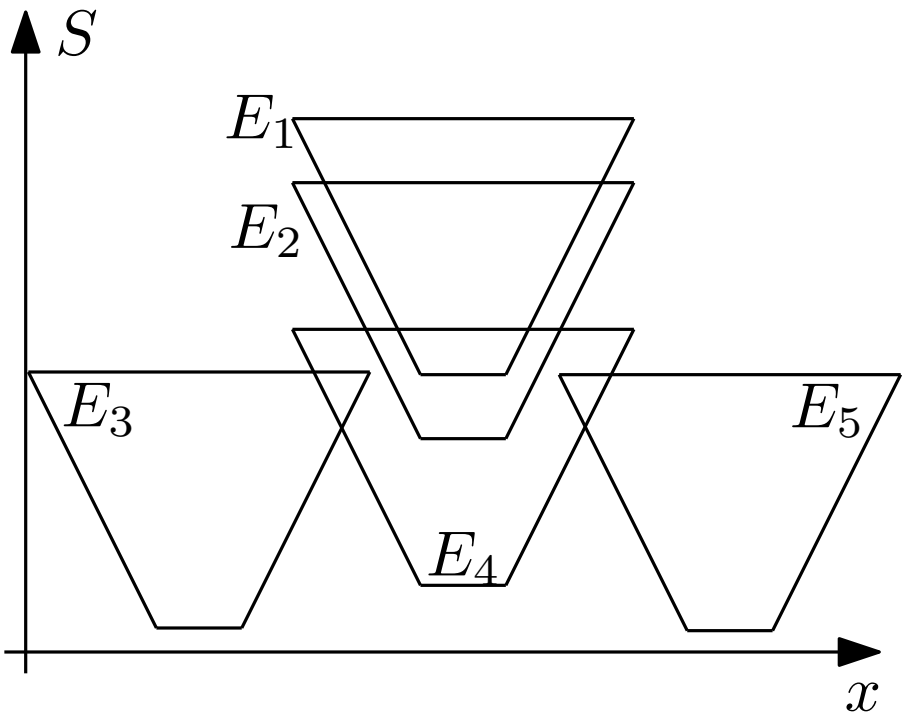
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



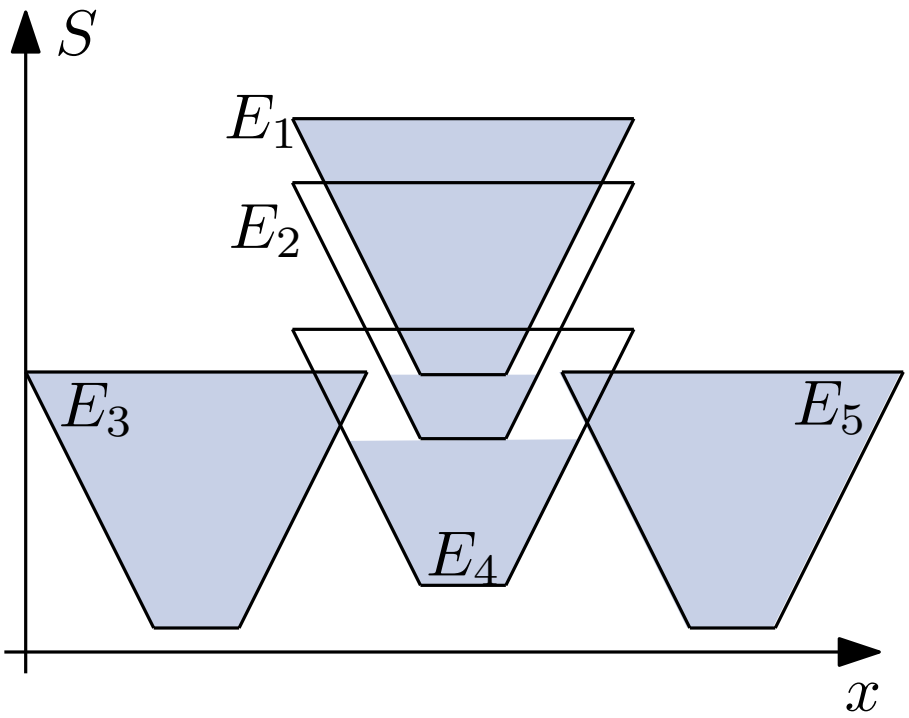
mit lexikografischer Ordnung



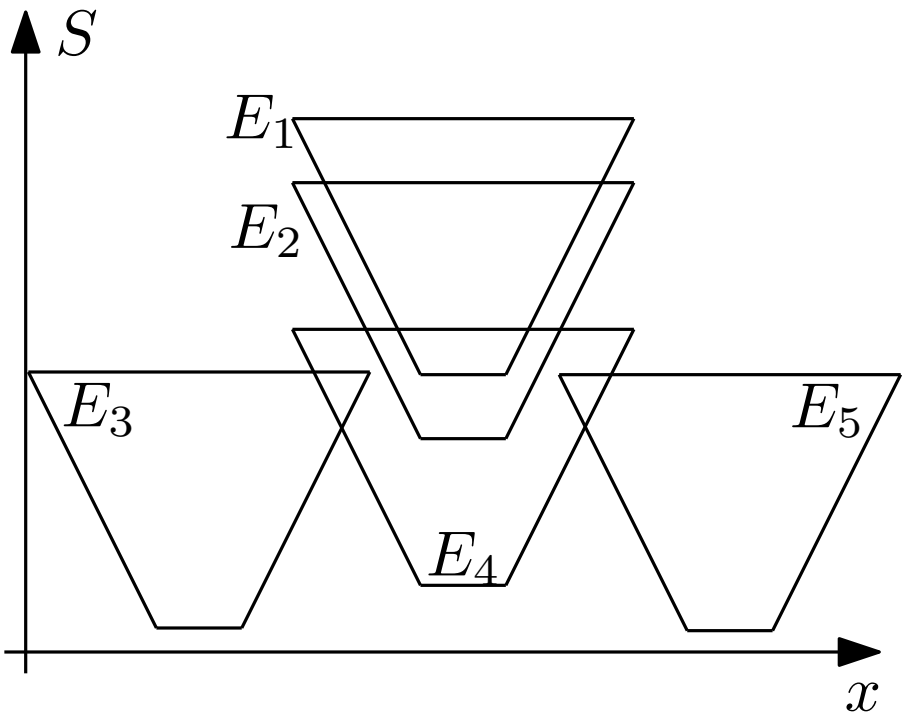
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



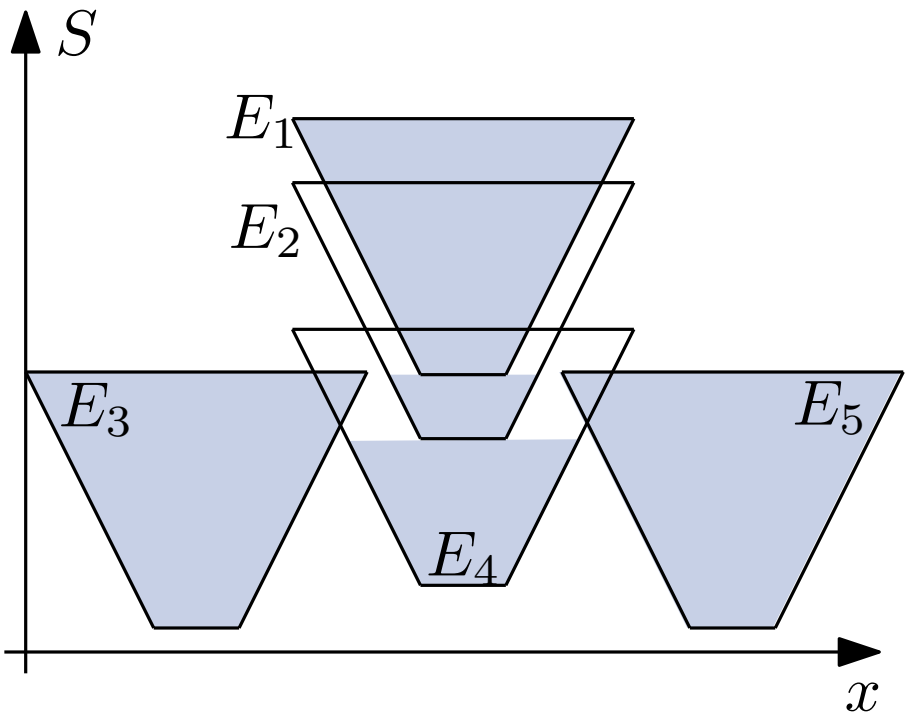
mit lexikografischer Ordnung



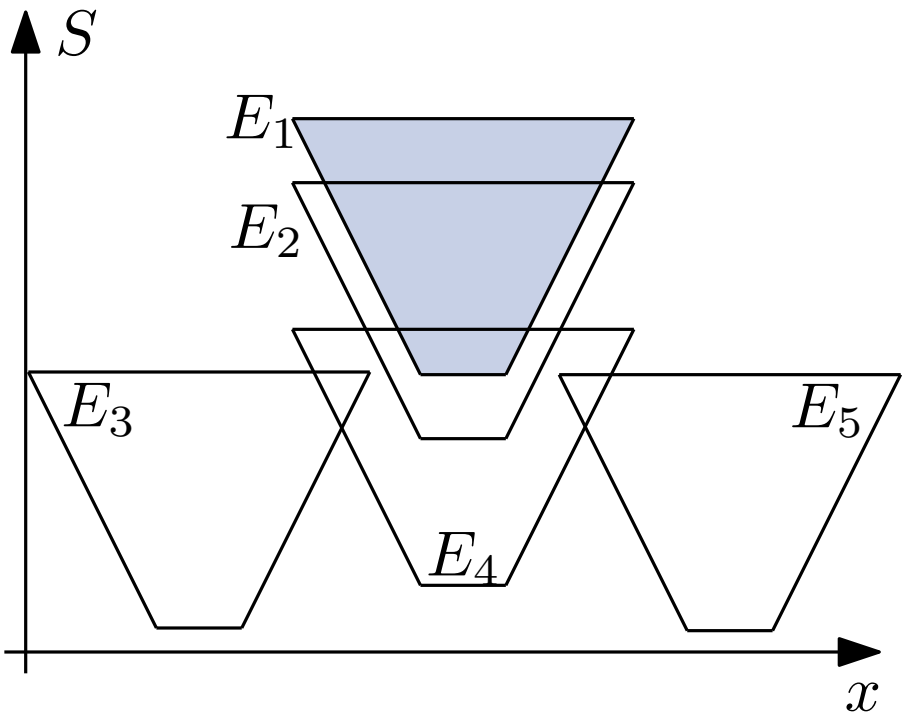
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



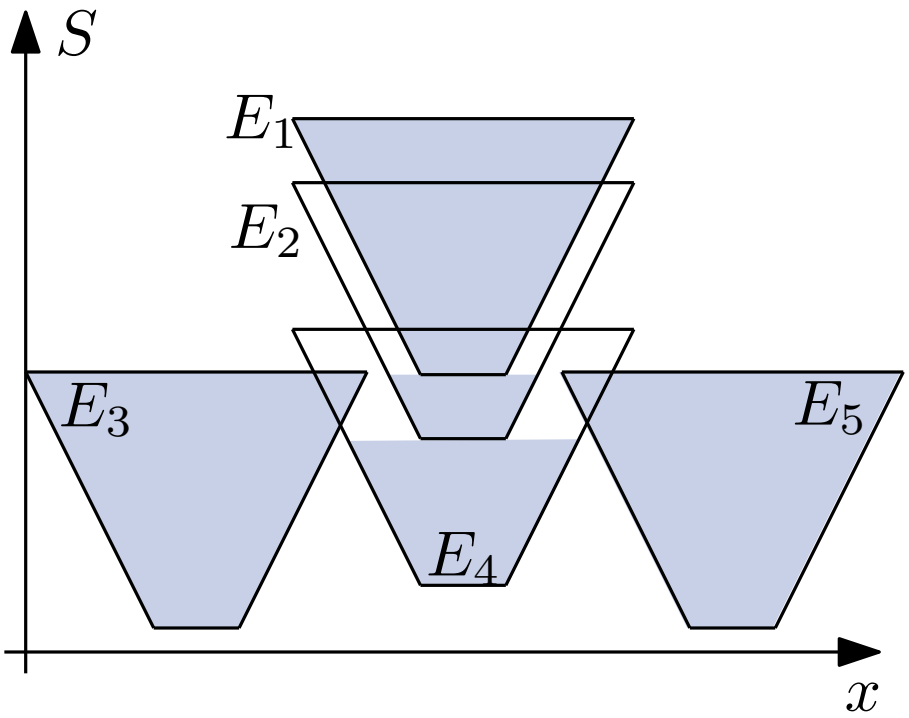
mit lexikografischer Ordnung



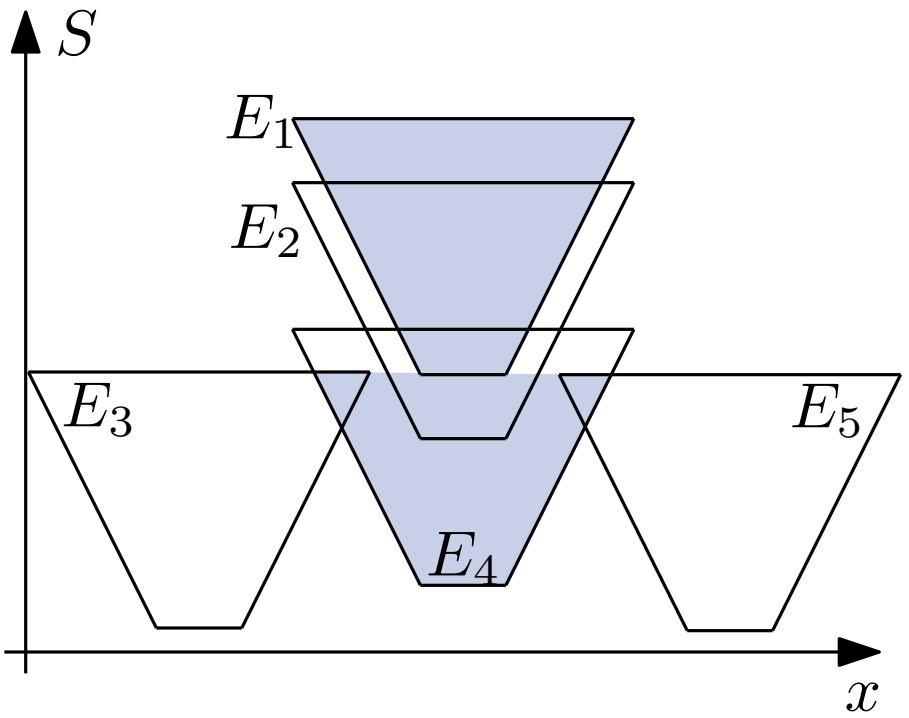
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



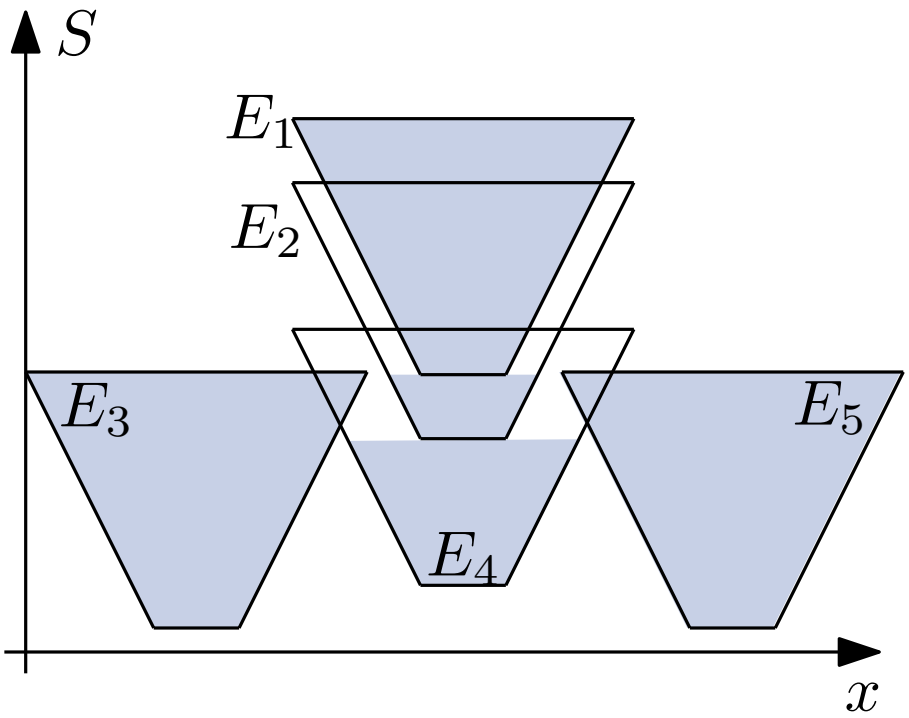
mit lexikografischer Ordnung



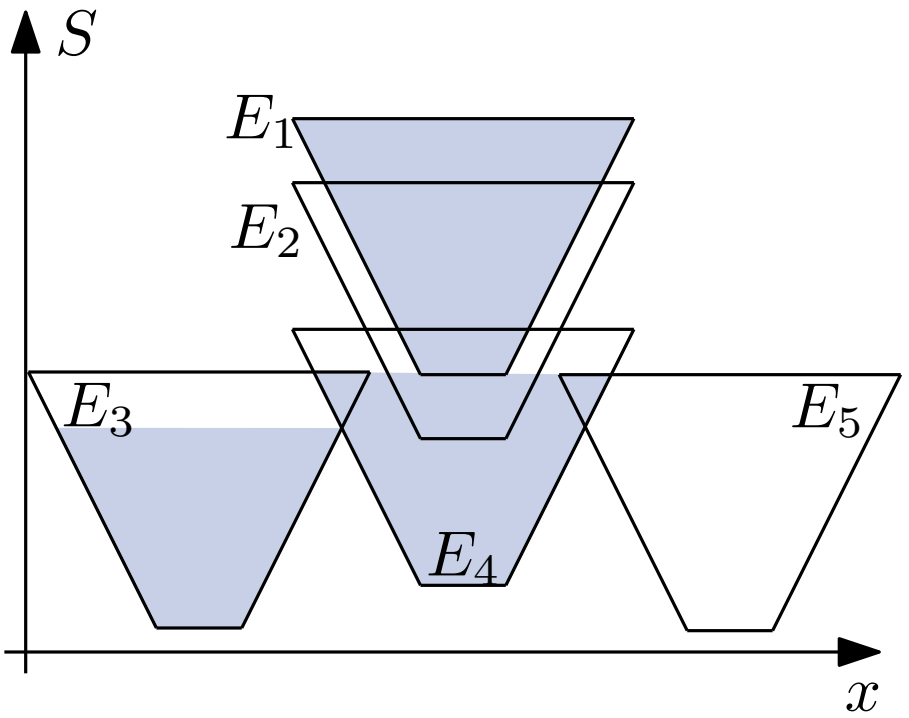
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



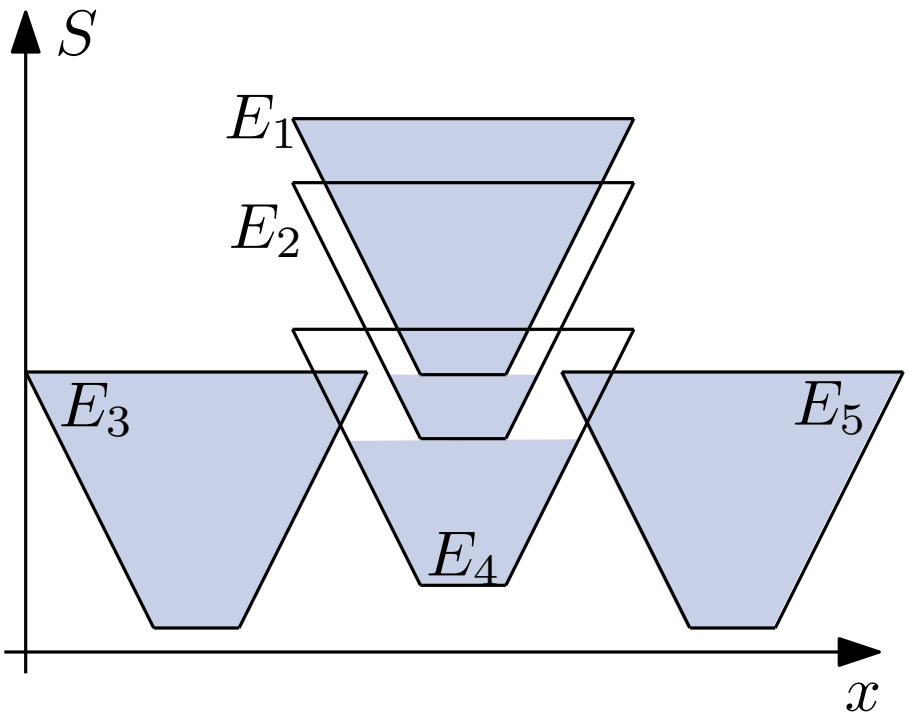
mit lexikografischer Ordnung



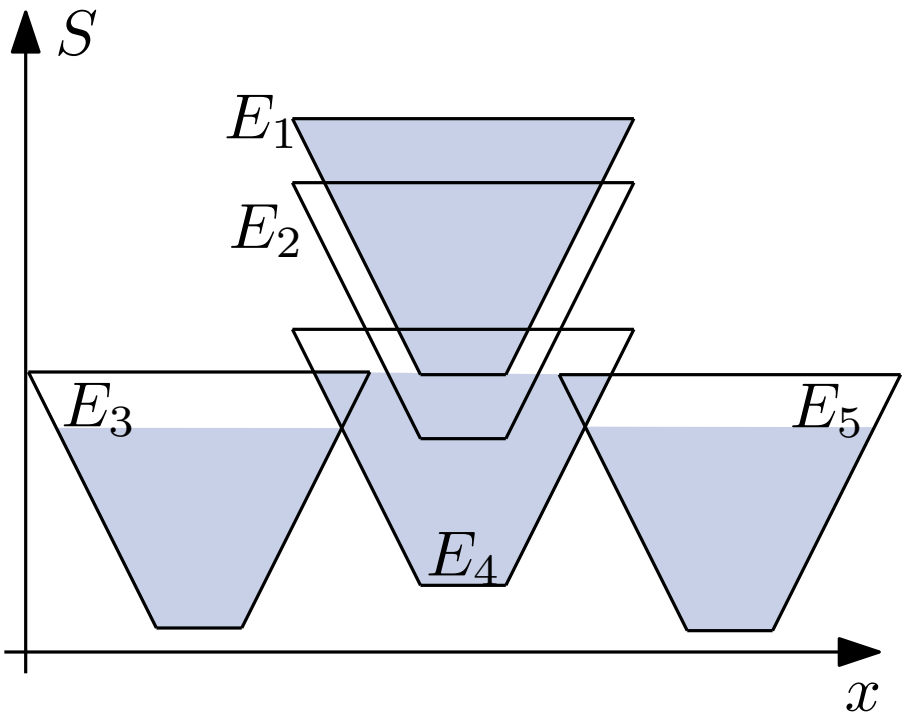
ohne lexikografischer Ordnung

Aufgabe 3

Was, wenn lexikografische Ordnung nicht verwendet wird?



mit lexikografischer Ordnung



ohne lexikografischer Ordnung

Aufgabe 1: 1D-Zooming

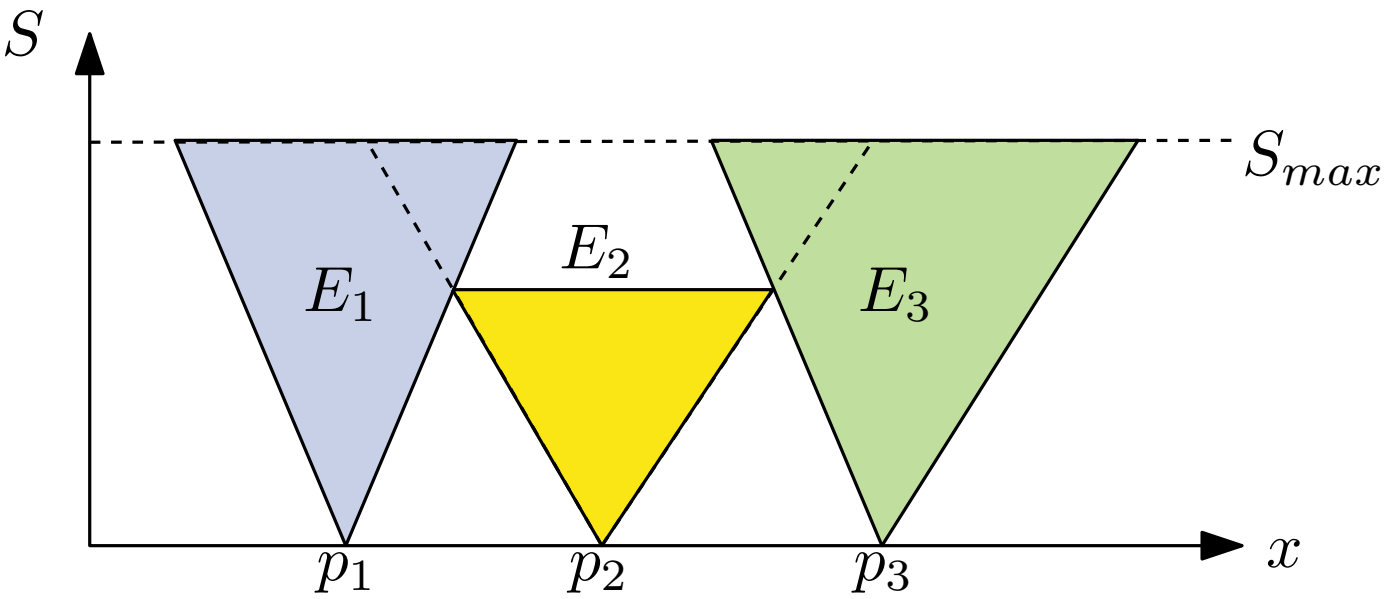
Aufgabe 1: 1D-Zooming

Aufgabe 1: 1D-Zooming

Aufgabe 1: 1D-Zooming

Aufgabe 1: 1D-Zooming

Aufgabe 1: 1D-Zooming



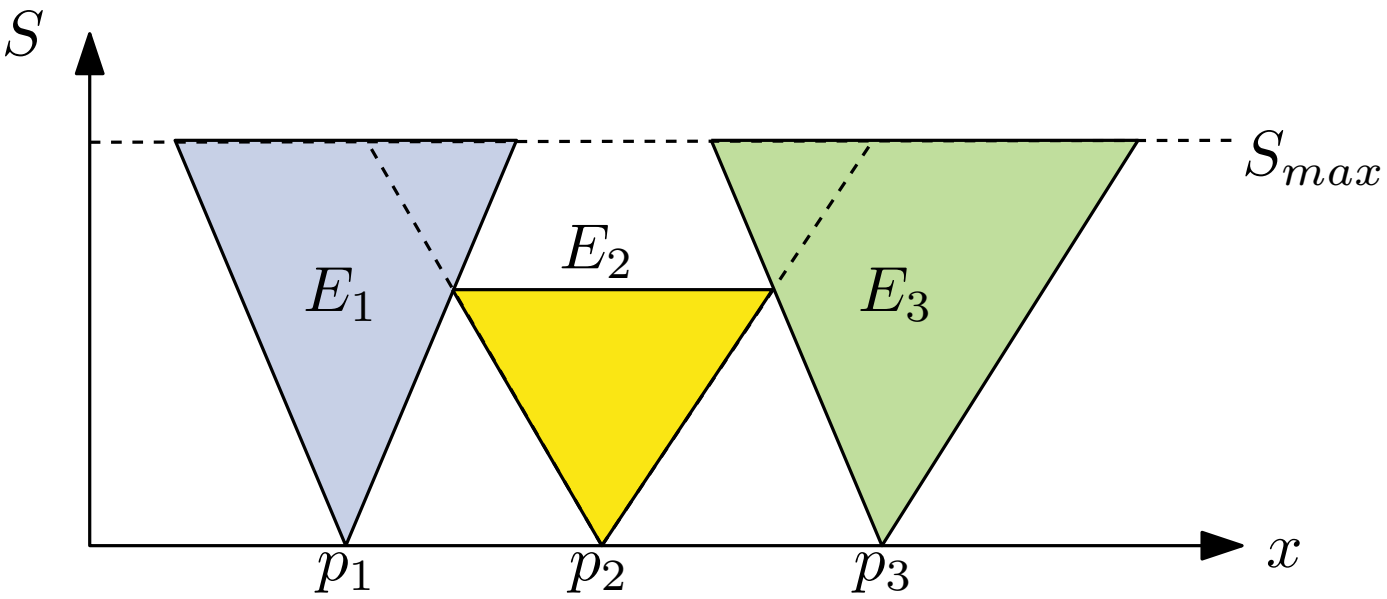
1D-Zooming:

Gegeben: Menge $\mathcal{E} = \{E_1, \ldots, E_n\}$ von Labeldreiecken für Punkte $p_1, \ldots, p_n$ mit verfügbaren Intervallen $(0, S_{max})$ für alle $E \in \mathcal{E}$.

Gesucht: Aktive Intervalle $(0, A_E)$ für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} A_E$ maximiert wird.

Annahme: Punkte sind sortiert von links nach rechts.

Aufgabe 1: 1D-Zooming



1D-Zooming:

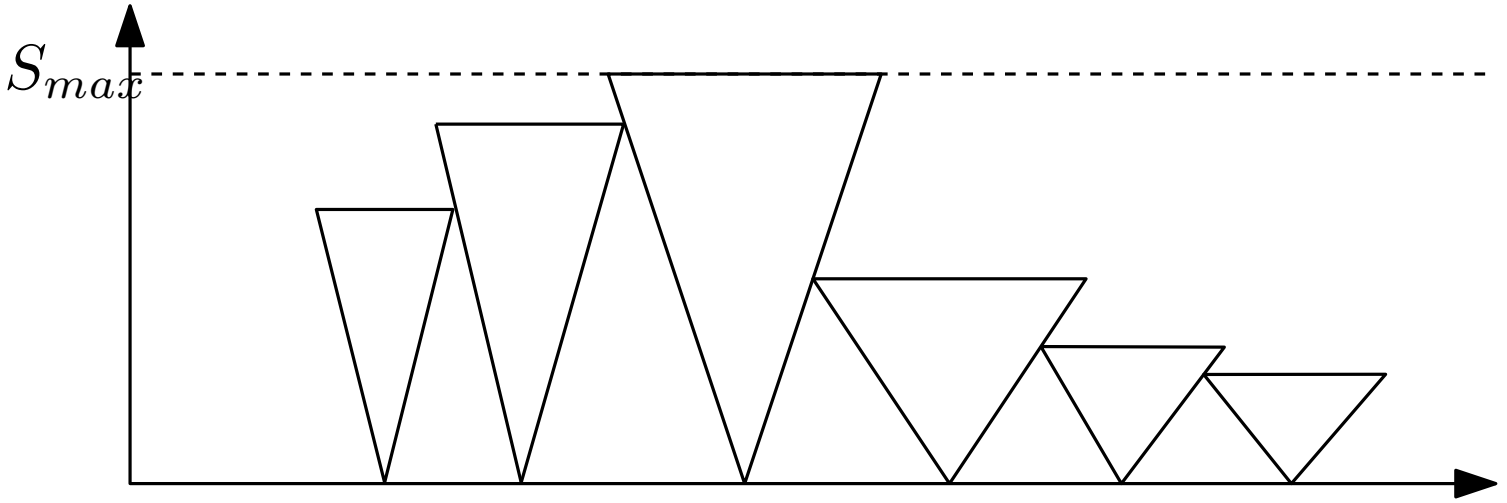
Gegeben: Menge $\mathcal{E} = \{E_1, \ldots, E_n\}$ von Labeldreiecken für Punkte $p_1, \ldots, p_n$ mit verfügbaren Intervallen $(0, S_{max})$ für alle $E \in \mathcal{E}$.

Gesucht: Aktive Intervalle $(0, A_E)$ für alle $E \in \mathcal{E}$, sodass $\sum_{E \in \mathcal{E}} A_E$ maximiert wird.

Annahme: Punkte sind sortiert von links nach rechts.

Aufgabe 1: 1D-Zooming – Dyn. Programm

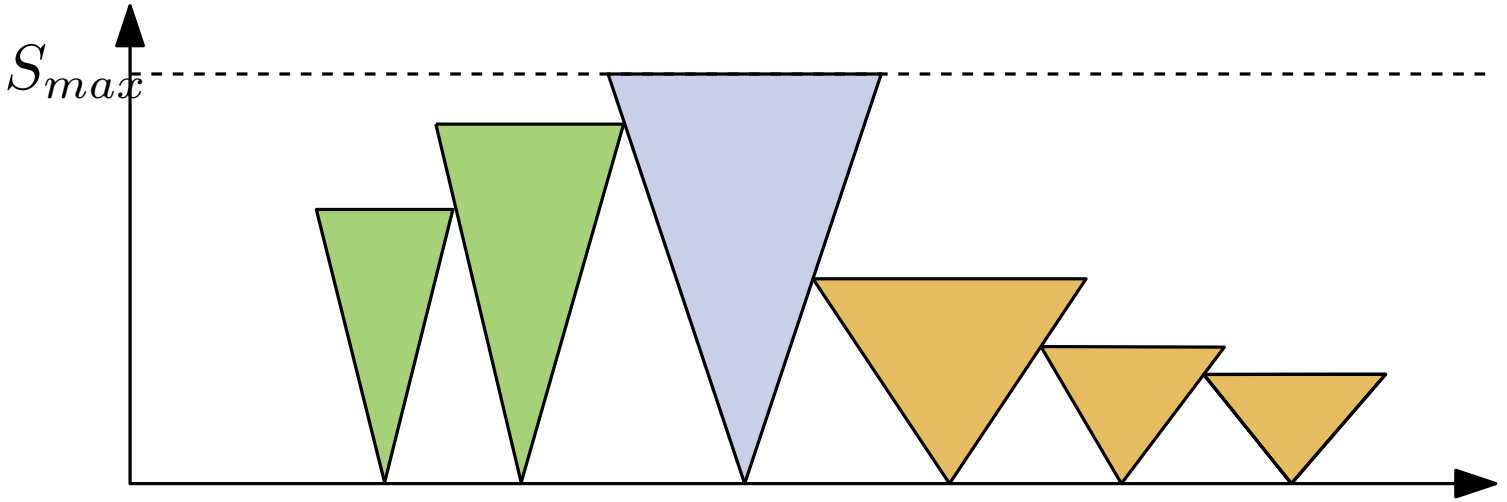
Betrachte optimale Lösung:



Höchste Dreieck teilt Instanz in zwei unabhängige Teile auf.

Aufgabe 1: 1D-Zooming – Dyn. Programm

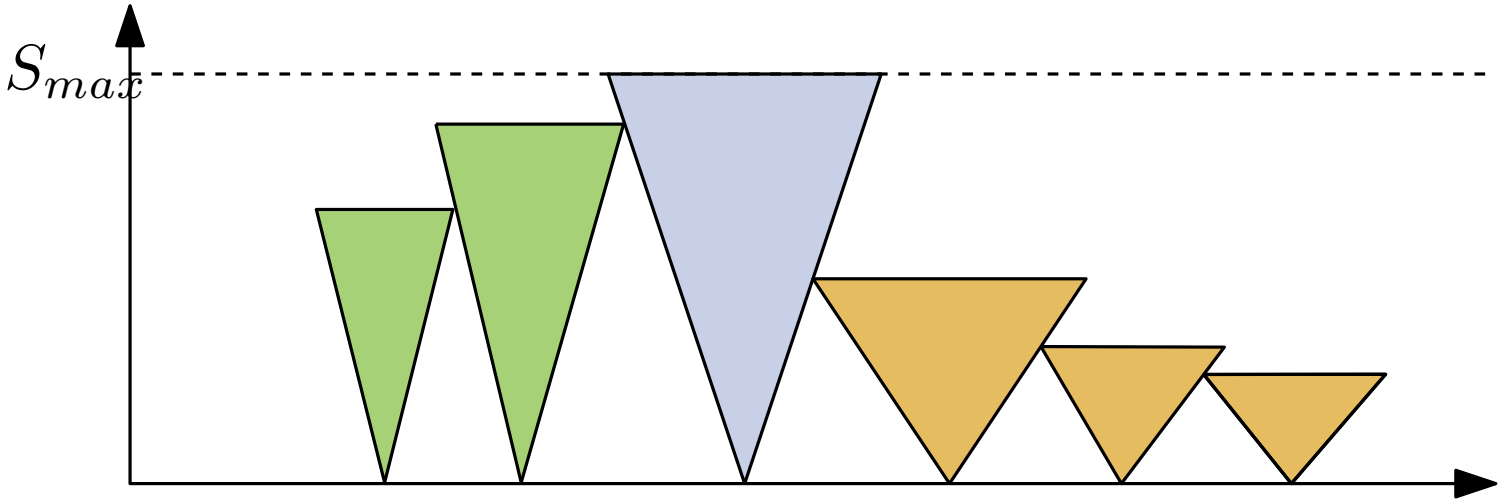
Betrachte optimale Lösung:



Höchste Dreieck teilt Instanz in zwei unabhängige Teile auf.

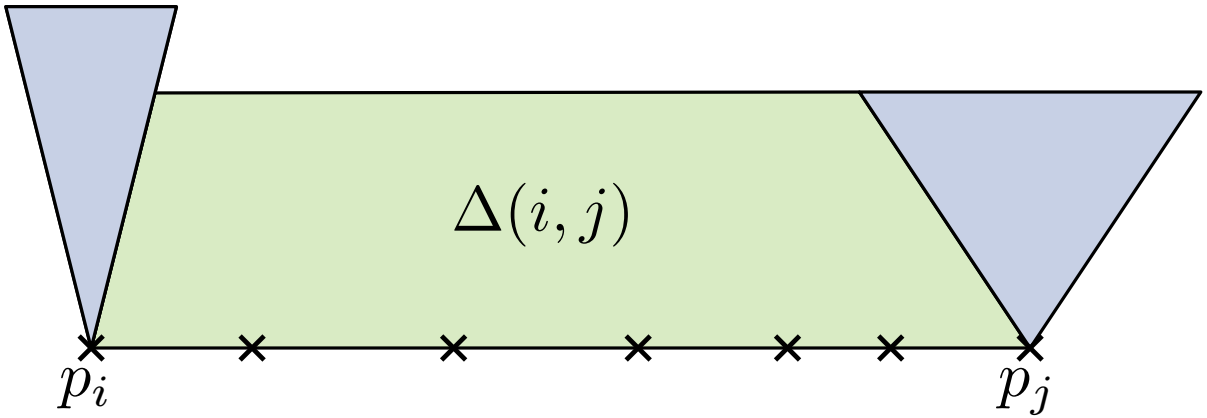
Aufgabe 1: 1D-Zooming – Dyn. Programm

Betrachte optimale Lösung:



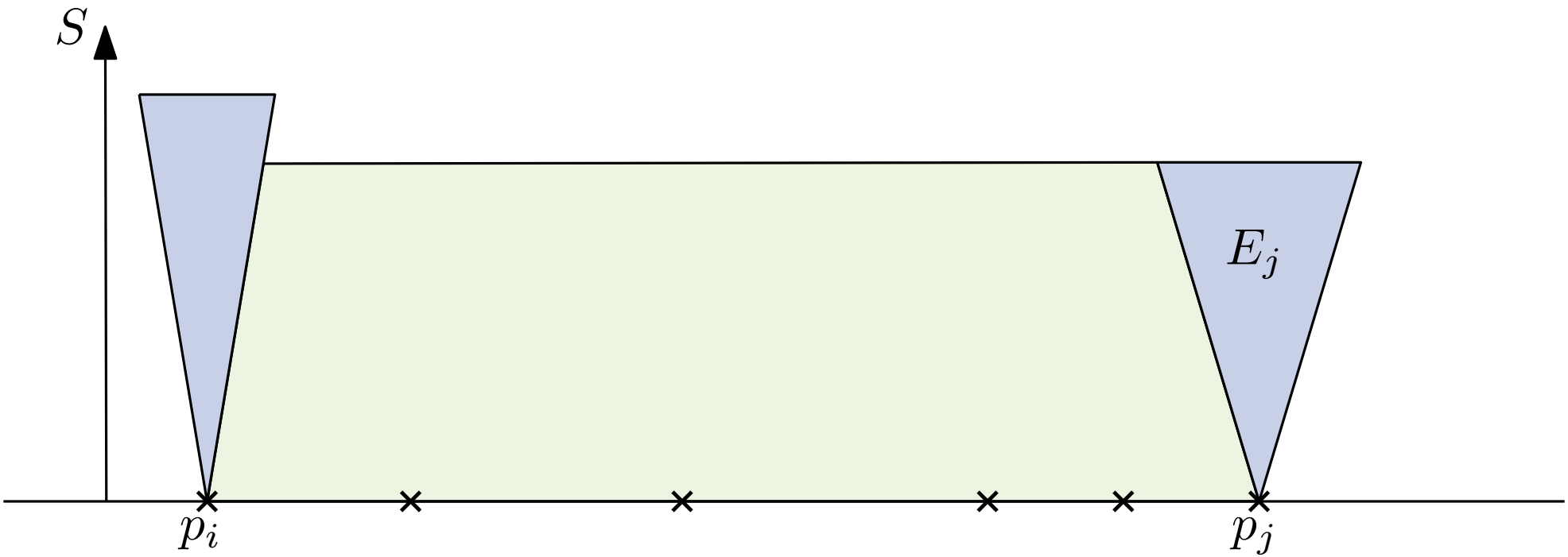
Höchste Dreieck teilt Instanz in zwei unabhängige Teile auf.

Idee: Begrenze zu lösende Instanz durch zwei bereits gesetzte Dreiecke:

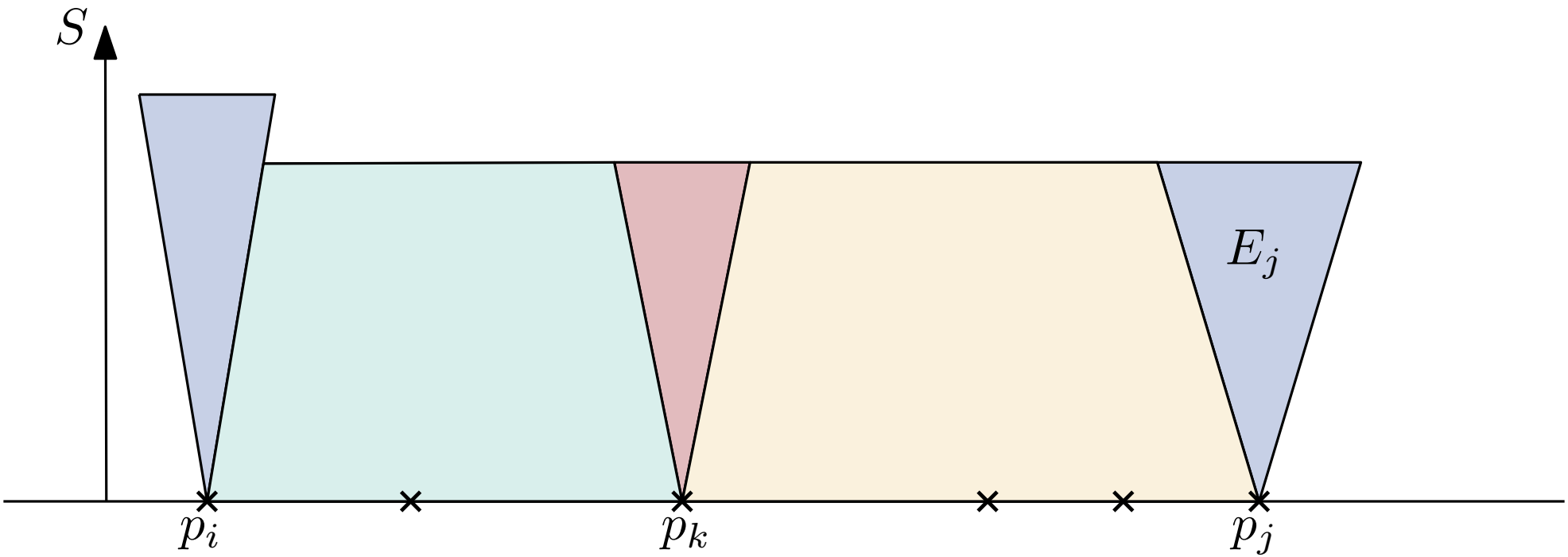


$A[i,j] =$ Optimale Lösung für $p_{i+1}, \ldots, p_{j-1}$ in $\Delta(i,j)$

Aufgabe 1: 1D-Zooming – Dyn. Programm

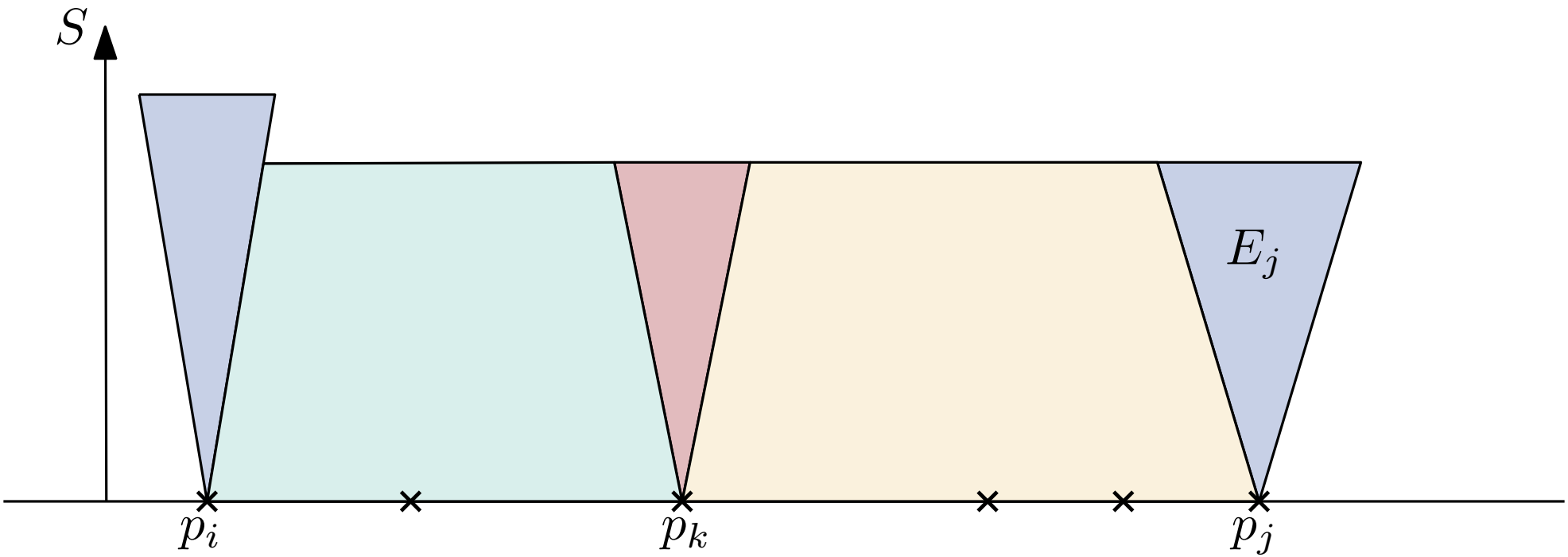


Aufgabe 1: 1D-Zooming – Dyn. Programm



In $A[i, j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i, j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

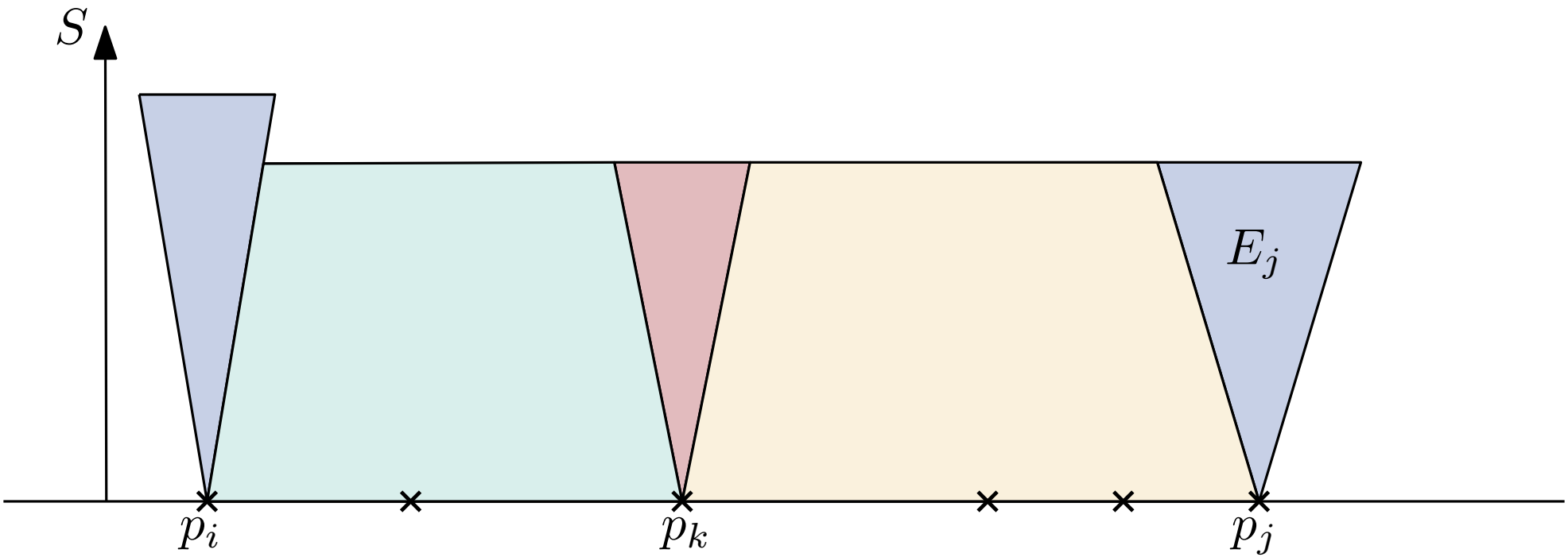
Aufgabe 1: 1D-Zooming – Dyn. Programm



In $A[i, j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i, j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck?

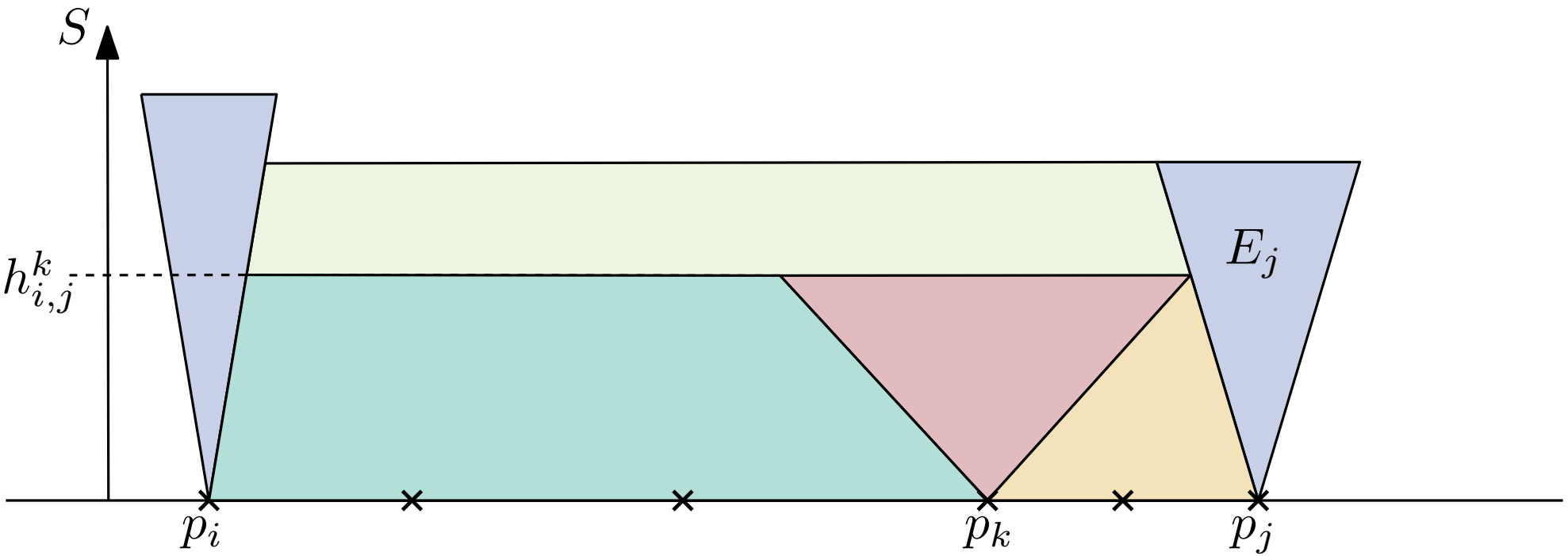
Aufgabe 1: 1D-Zooming – Dyn. Programm



In $A[i, j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i, j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck? Probiere alle Möglichkeiten aus.

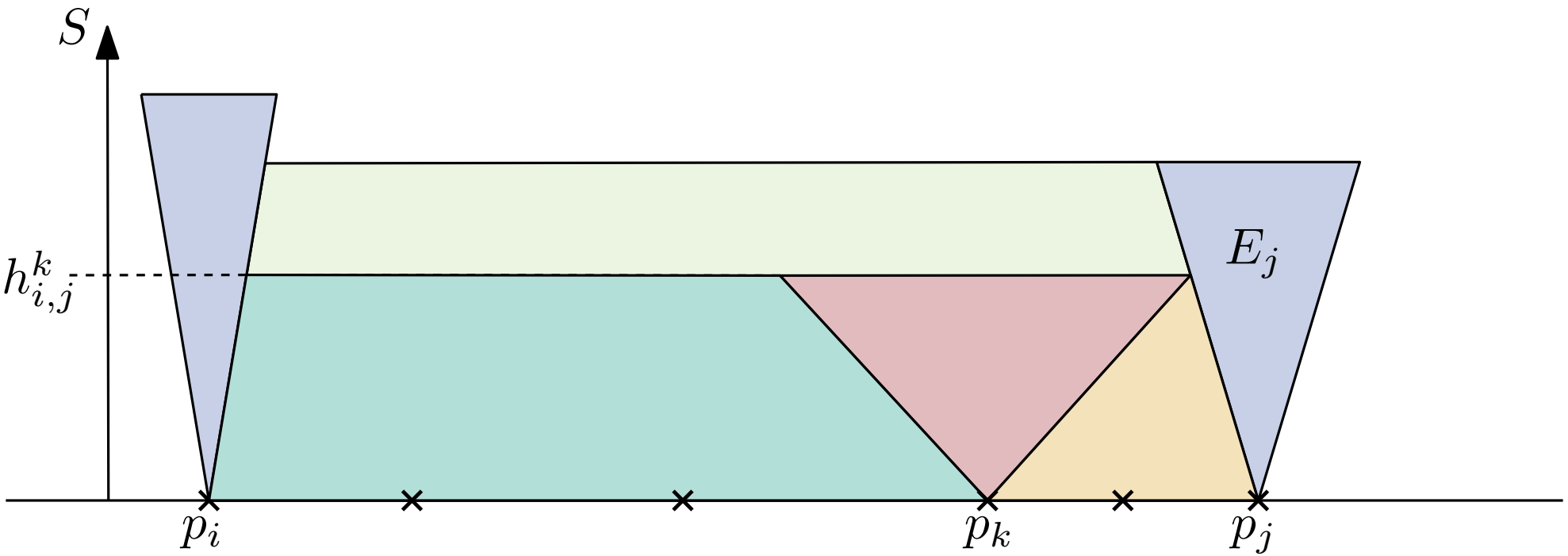
Aufgabe 1: 1D-Zooming – Dyn. Programm



In $A[i,j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i,j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck? Probiere alle Möglichkeiten aus.

Aufgabe 1: 1D-Zooming – Dyn. Programm

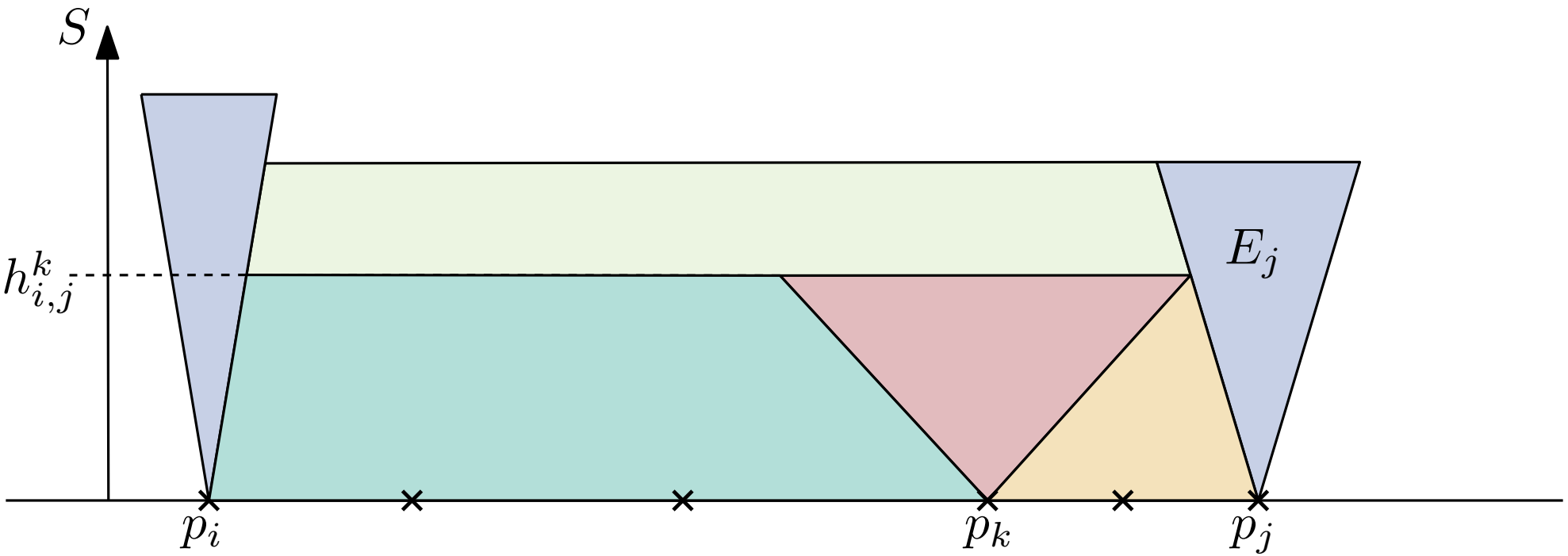


In $A[i,j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i,j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck? Probiere alle Möglichkeiten aus.

$$A[i,j] = \max\{A[i,k] + h_k^{i,j} + A[k,j] \mid i+1 \leq k \leq j-1\}$$

Aufgabe 1: 1D-Zooming – Dyn. Programm



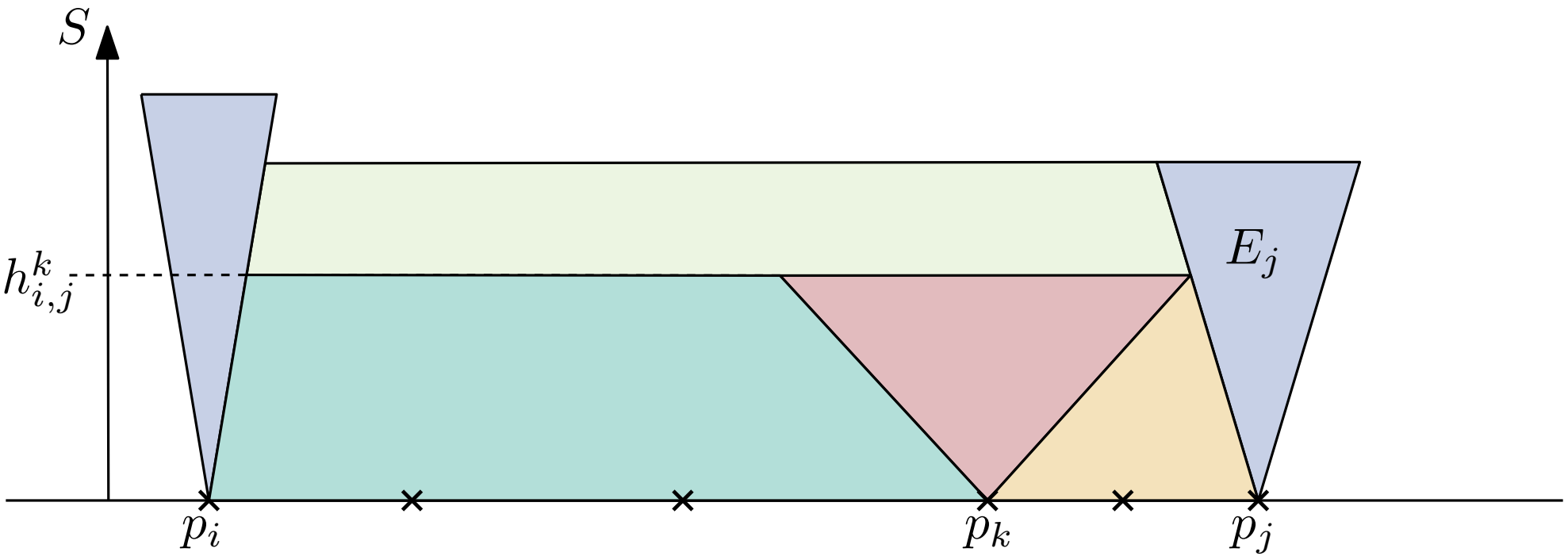
In $A[i,j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i,j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck? Probiere alle Möglichkeiten aus.

$$A[i,j] = \max\{A[i,k] + h_k^{i,j} + A[k,j] \mid i+1 \leq k \leq j-1\}$$

Laufzeit und Speicherbedarf?

Aufgabe 1: 1D-Zooming – Dyn. Programm



In $A[i,j]$ gibt es ein Dreieck, das eine Seite von $\Delta(i,j)$ berührt, die nicht die untere Seite ist $\rightarrow$ unterteilt Instanz.

Welches Dreieck? Probiere alle Möglichkeiten aus.

$$A[i,j] = \max\{A[i,k] + h_k^{i,j} + A[k,j] \mid i+1 \leq k \leq j-1\}$$

$O(n^2)$ Speicher und $O(n^3)$ Laufzeit