Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Chair for
INFORMATICS I
Efficient Algorithms and
Knowledge-Based Systems

Institute for Informatics

Approximationsalgorithmen

Set Cover
und
Shortest SuperString

2. Vorlesung

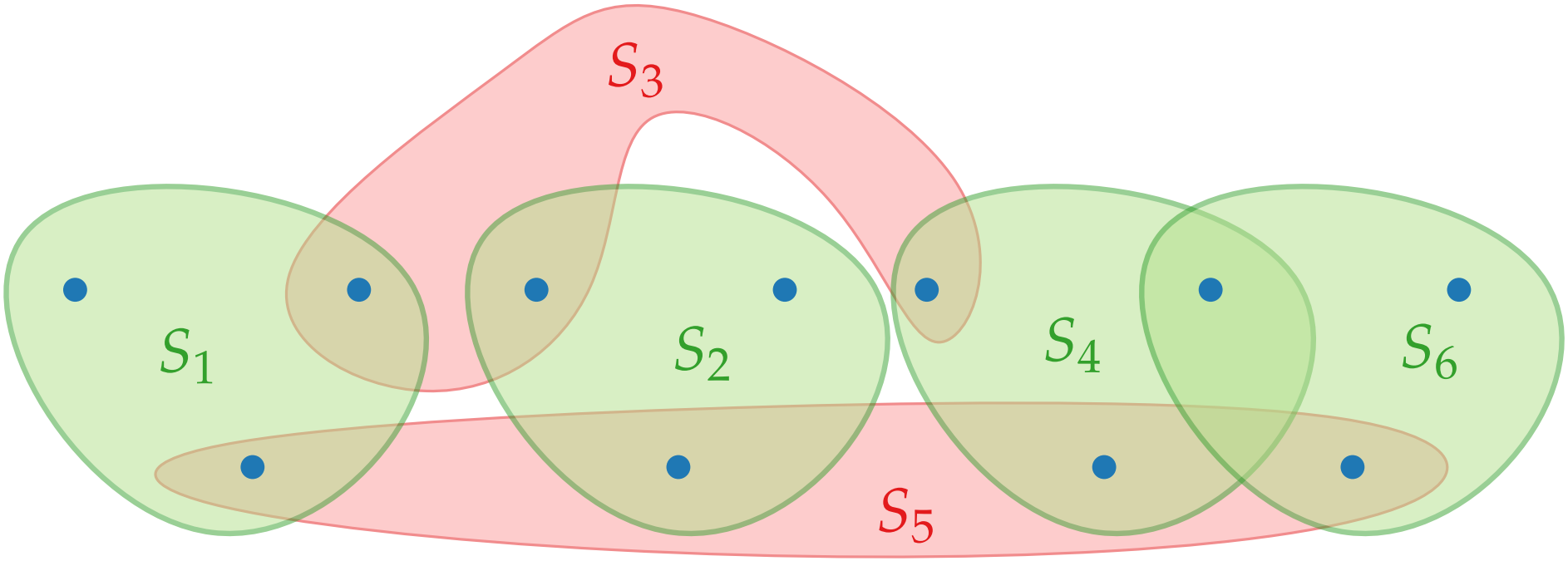
Philipp Kindermann

Wintersemester 2019/20

SetCover (card.)

Gegeben sei eine **Grundmenge** U und eine Familie $\mathcal{S}$ von **Teilmengen** von U mit $\bigcup \mathcal{S} = U$.

Gesucht ist eine **Überdeckung** $\mathcal{S}' \subseteq \mathcal{S}$ von U (d.h. $\bigcup \mathcal{S}' = U$) minimaler Kardinalität.



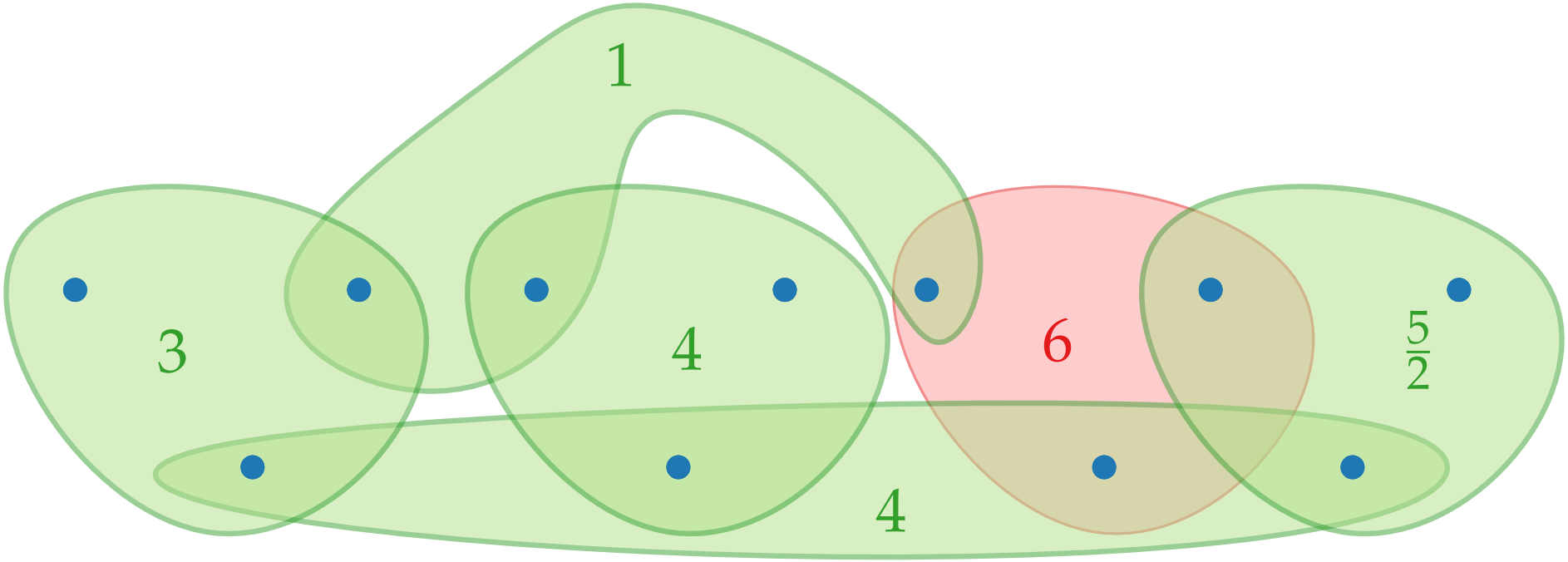
SetCover (allgemein)

Gegeben sei eine **Grundmenge** U und eine Familie $\mathcal{S}$ von **Teilmengen** von U mit $\bigcup \mathcal{S} = U$.

Jedes $S \in \mathcal{S}$ hat Kosten $c(S) > 0$.

Gesucht ist eine **Überdeckung** $\mathcal{S}' \subseteq \mathcal{S}$ von U (d.h. $\bigcup \mathcal{S}' = U$) minimaler ~~Kardinalität~~.

Kosten $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c(S)$.

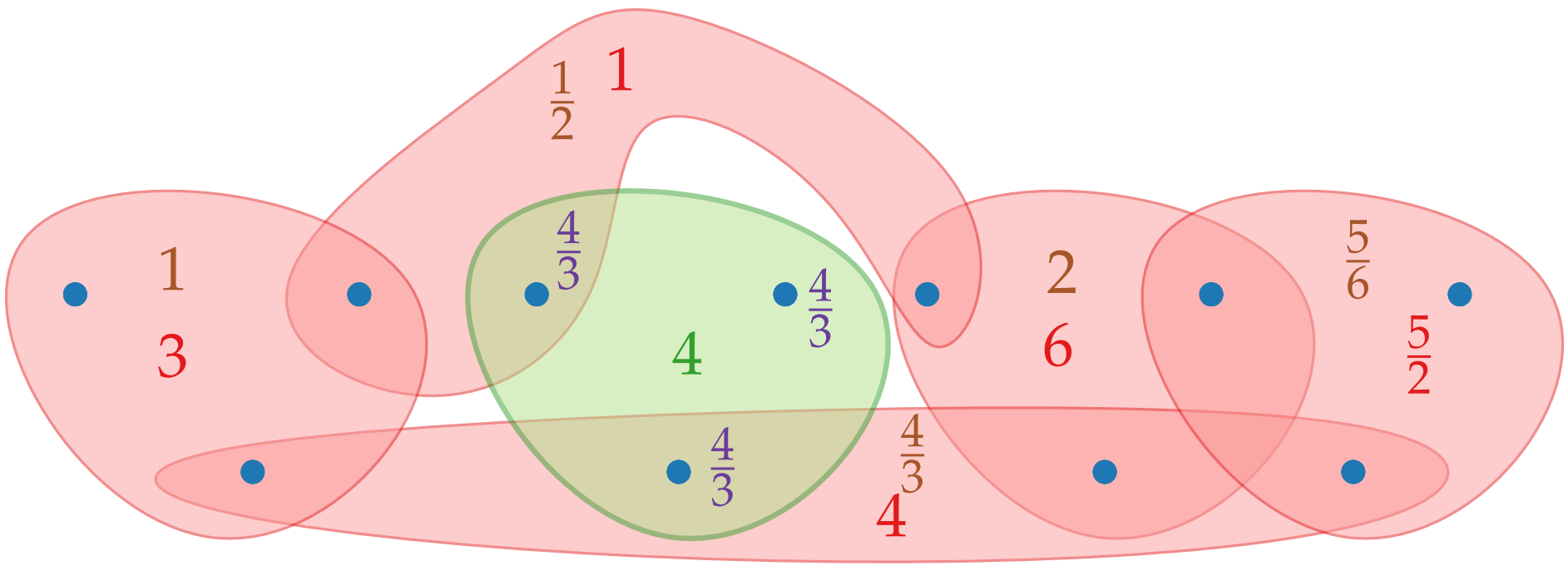


Iteratives „Einkaufen“ von Elementen

Was sind die wirklichen Kosten für eine Menge?

Menge mit k Elementen und Kosten c hat Einheitspreis $\frac{c}{k}$.

Was passiert, wenn wir eine Menge kaufen?
Fixiere Preis der gekauften Elemente und berechne Einheitspreise neu.



Iteratives „Einkaufen“ von Elementen

Was sind die wirklichen Kosten für eine Menge?

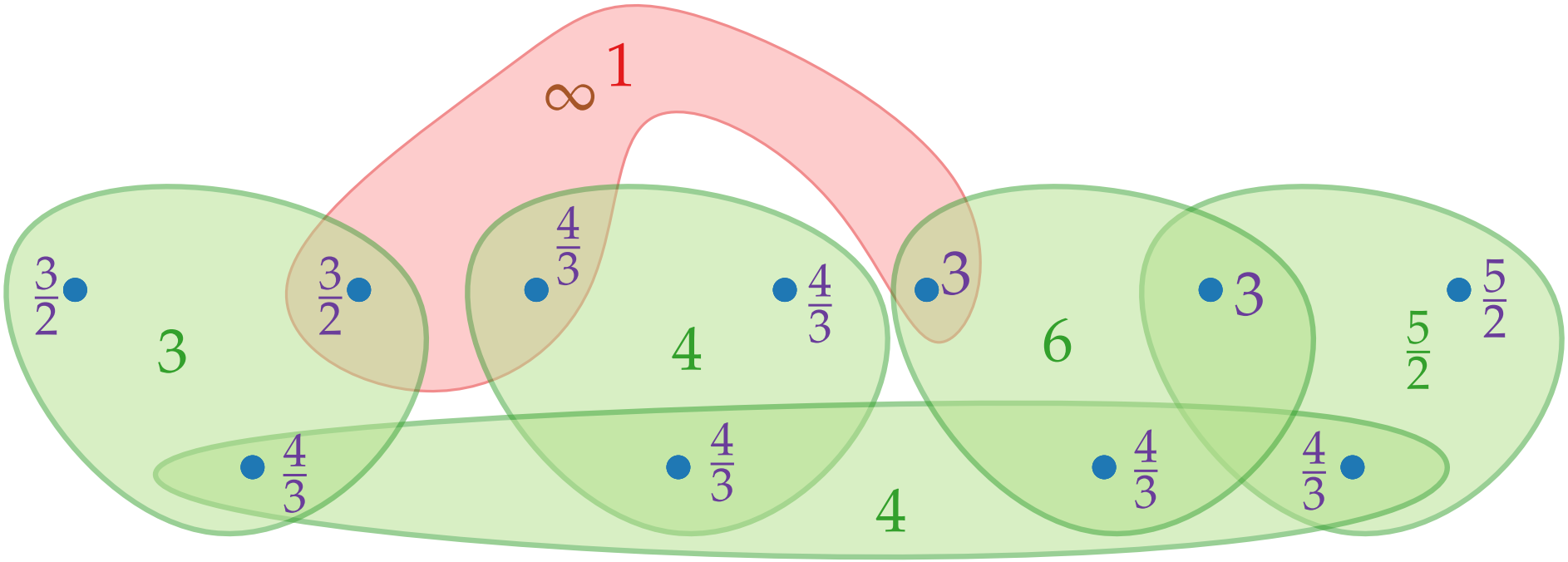
Menge mit k Elementen und Kosten c hat Einheitspreis $\frac{c}{k}$.

Was passiert, wenn wir eine Menge kaufen?

Fixiere Preis der gekauften Elemente und berechne Einheitspreise neu.

Gesamtkosten: $\sum_{u \in U} \text{preis}(u)$

Greedy: Wähle immer Menge mit kleinstem Einheitspreis.



Greedy für SetCover

GreedySetCover$(U, \mathcal{S}, c)$

$C \leftarrow \emptyset$
$\mathcal{S}' \leftarrow \emptyset$
while $C \neq U$ **do**
  $S \leftarrow$ Menge aus $\mathcal{S}$, die $\frac{c(S)}{|S \setminus C|}$ minimiert
  foreach $u \in S \setminus C$ **do**
    $\text{preis}(u) \leftarrow \frac{c(S)}{|S \setminus C|}$
  $C \leftarrow C \cup S$
  $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S\}$
return $\mathcal{S}'$ `// Überdeckung von` U

Analyse

Satz. GreedySetCover ist ein Faktor-H_k-Approximationsalg. für SetCover. Hierbei ist k die Kardinalität der größten Menge in $\mathcal{S}$ und $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Sei $S \in \mathcal{S}$ und seien $u_1, \ldots, u_\ell$ die Elemente von S in der Reihenfolge, in der sie von GreedySetCover überdeckt („gekauft“) werden. Dann gilt $\text{preis}(u_j) \leq c(S)/(\ell - j + 1)$.

Beweis. Alg. wählt $u_j \Rightarrow$

- $j - 1$ Elemente von S bereits gewählt
- $\ell - j + 1$ Elemente von S noch nicht gewählt
- Einheitskosten für S: $c(S)/(\ell - j + 1)$

Analyse

Satz. GreedySetCover ist ein Faktor-H_k-Approximationsalg. für SetCover. Hierbei ist k die Kardinalität der größten Menge in $\mathcal{S}$ und $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Sei $S \in \mathcal{S}$ und seien $u_1, \ldots, u_\ell$ die Elemente von S in der Reihenfolge, in der sie von GreedySetCover überdeckt („gekauft“) werden. Dann gilt $\text{preis}(u_j) \leq c(S)/(\ell - j + 1)$.

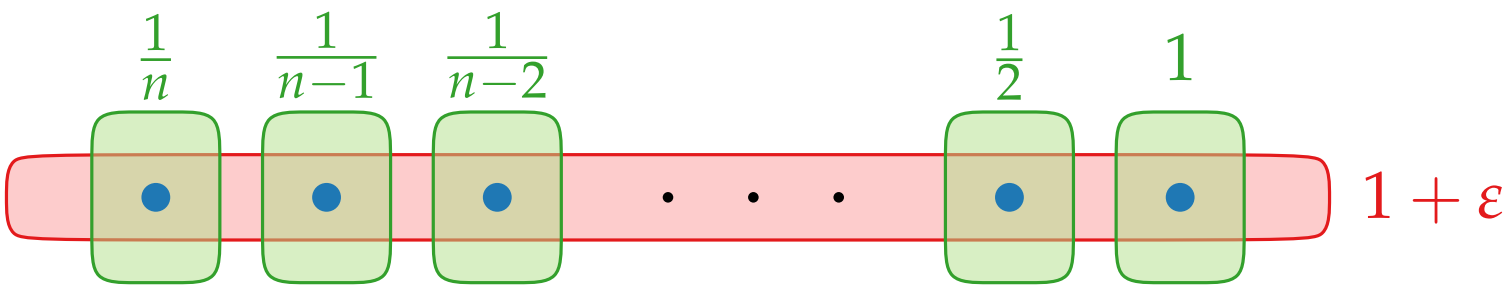
Lemma. $\text{preis}(S) := \sum_{i=1}^{\ell} \text{preis}(u_i) \leq c(S) \cdot H_\ell$.

Beweis. Sei $\{S_1, \ldots, S_m\}$ optimale Lsg. $\text{OPT} = \sum_{i=1}^{m} c(S_i)$

$$\text{preis}(U) = \sum_{u \in U} \text{preis}(u) \leq \sum_{i=1}^{m} \text{preis}(S_i)$$
$$\leq \sum_{i=1}^{m} c(S_i) \cdot H_k = \text{OPT} \cdot H_k$$

Analyse scharf?

Satz. GreedySetCover ist ein Faktor-H_k-Approximationsalg. für SetCover. Hierbei ist k die Kardinalität der größten Menge in $\mathcal{S}$ und $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



$\text{preis}(U) = H_n$ $\qquad$ $\text{OPT} = 1 + \varepsilon$

besser?

SetCover lässt sich nicht mit Faktor $(1 - o(1)) \cdot \log(n)$ approximieren (außer P=NP)

ShortestSuperString (SSS)

Gegeben sei eine Menge $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ von Strings über einem endlichen Alphabet Σ. Gesucht ist ein **kürzester String** s (*Superstring*), der jeden String s_i, $i = 1, \ldots, n$, als Teilstring enthält.

Beispiel. $U := \{cbaa, abc, bcb\}$

„überdeckt“ alle Strings aus U

Ohne Einschränkung: Kein String s_i ist Teilstring eines anderen Strings s_j

$abcbaa$

abc
$\;\;bcb$
$\;\;\;cbaa$

SSS als SetCover Problem

SetCover Instanz: Grundmenge U, Teilmengen $\mathcal{S}$, Kosten c.

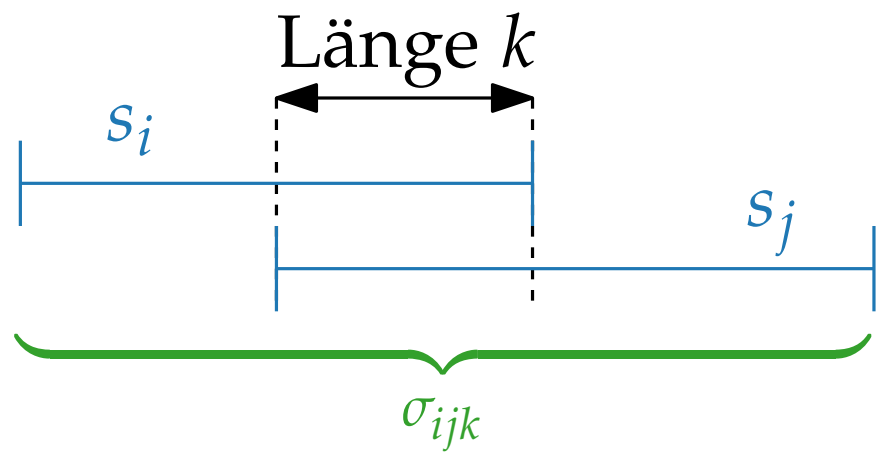
Grundmenge $U := \{s_1, \ldots, s_n\}$

Ein String σ_{ijk} hat Präfix s_i, Suffix s_j, wobei sich s_i und s_j in k Zeichen *überlappen*.

s_i : cabab $\quad s_j$: ababc

```
    cabab                cabab
     ababc                ababc
```

σ_{ij2} : cababababc $\quad \sigma_{ij4}$: cababc



$S(\sigma_{ijk}) = \{s \in U \mid s$ Teilstring von $\sigma_{ijk}\}$ enthält die Elemente der Grundmenge, die durch σ_{ijk} überdeckt werden.

$c\left(S(\sigma_{ijk})\right) = |\sigma_{ijk}|$ (Anzahl der Zeichen in σ_{ijk})

$\mathcal{S} = \{S(\sigma_{ijk}) \mid k > 0\}$ ($i = j$ erlaubt)

Beziehung SSS und SetCover

Lemma. Sei $\mathrm{OPT}_{\mathrm{SSS}}$ die Länge des kürzesten Superstring von U und $\mathrm{OPT}_{\mathrm{SC}}$ die kleinsten Kosten der dazugehörigen SetCover Instanz. Dann gilt:

$$\mathrm{OPT}_{\mathrm{SSS}} \leq \mathrm{OPT}_{\mathrm{SC}}$$

Beweis.

Betrachte optimales Set Cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ von U.

$s := \pi_1 \circ \cdots \circ \pi_k$ ist ein Superstring von U mit Länge $\sum_{i=1}^{k} |\pi_i| = \mathrm{OPT}_{\mathrm{SC}}$

Also $\mathrm{OPT}_{\mathrm{SSS}} \leq |s| = \mathrm{OPT}_{\mathrm{SC}}$

Beziehung SSS und SetCover

Lemma. $\mathrm{OPT}_{\mathrm{SC}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

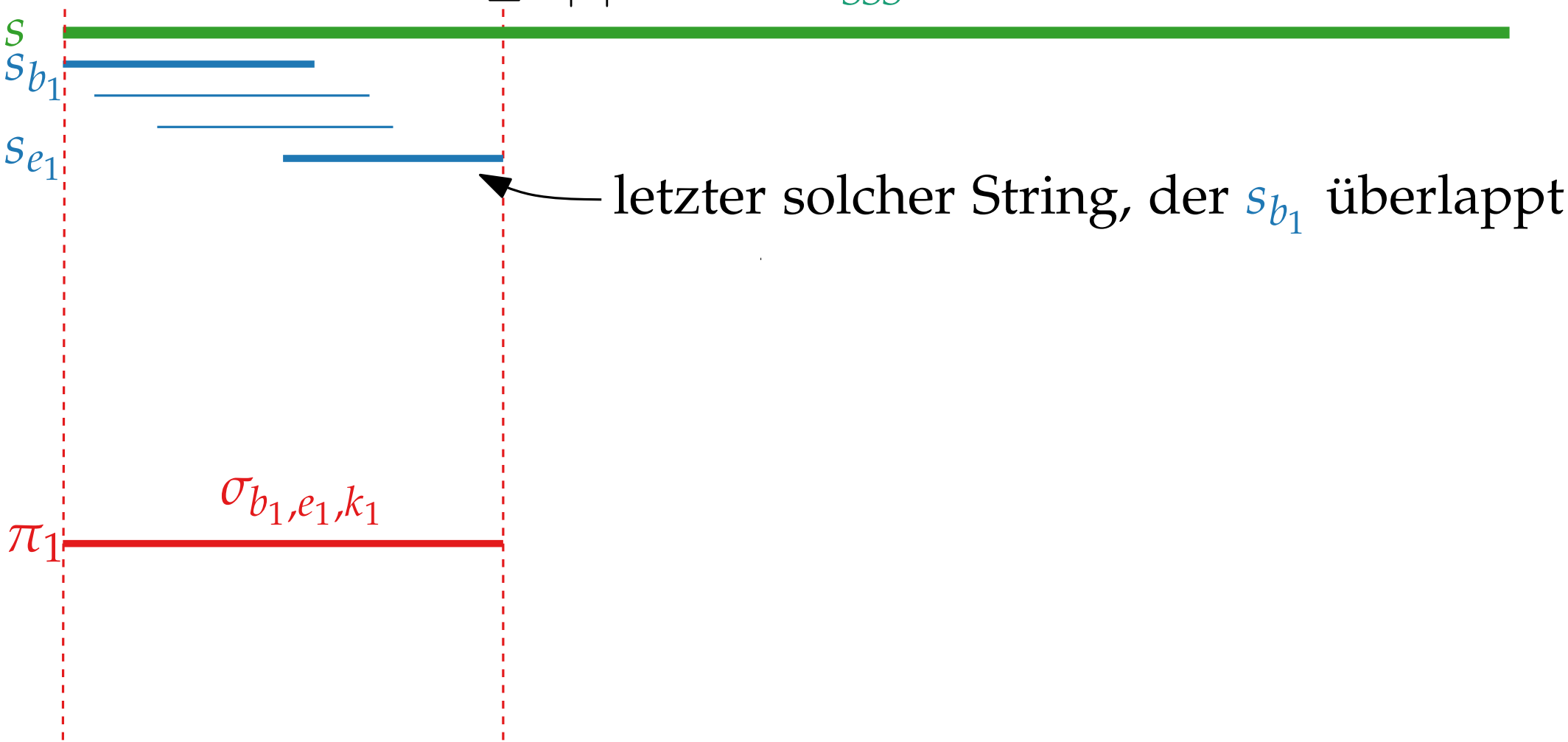
Beweis. Betrachte optimalen Superstring s.
Konstruiere Überdeckung mit
Kosten $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

linkestes Auftreten eines Strings $s_{b_1} \in U$

Beziehung SSS und SetCover

Lemma. $\mathrm{OPT}_{\mathrm{SC}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

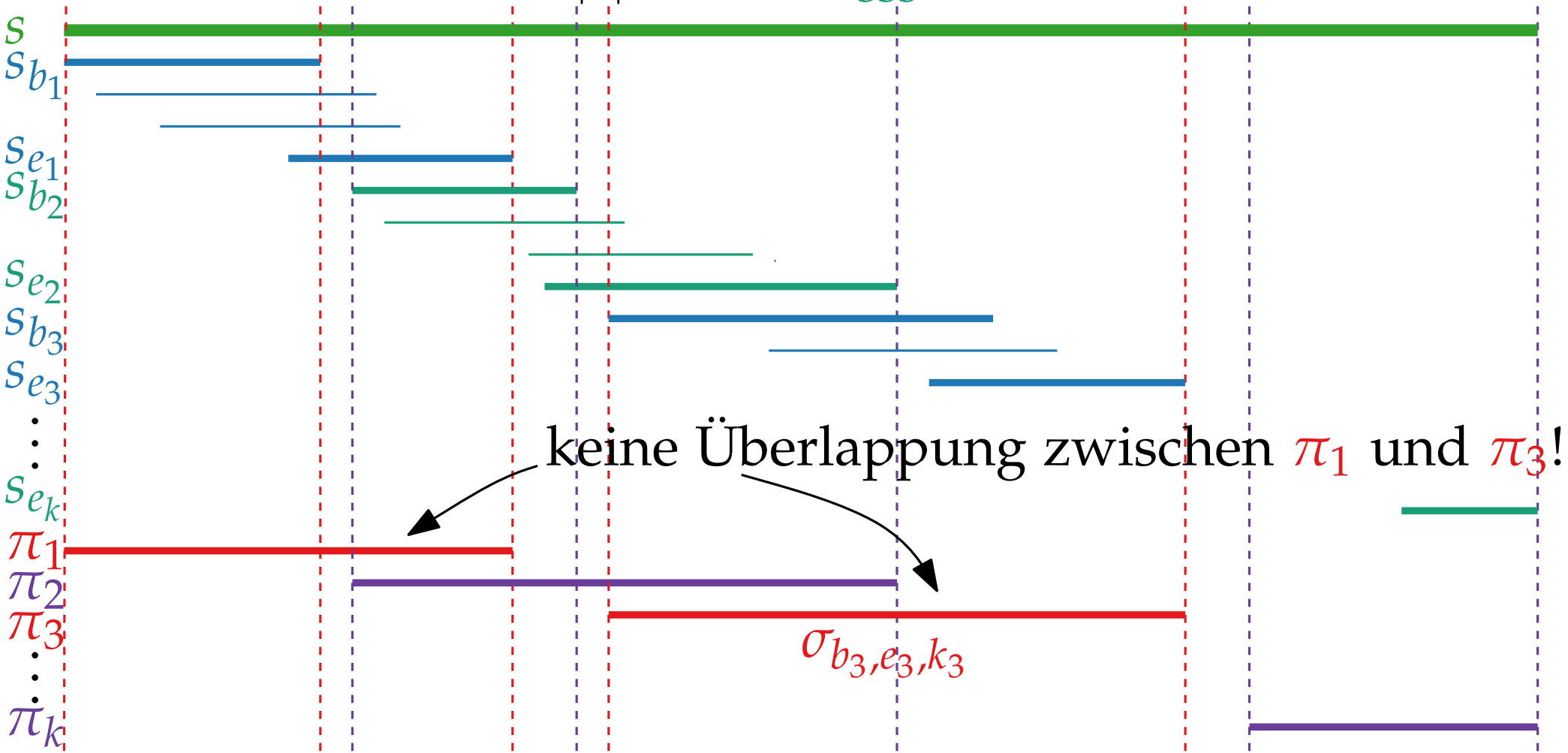
Beweis. Betrachte optimalen Superstring s.
Konstruiere Überdeckung mit
Kosten $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Beziehung SSS und SetCover

Lemma. $\mathrm{OPT}_{\mathrm{SC}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

Beweis. Betrachte optimalen Superstring s.
Konstruiere Überdeckung mit
Kosten $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Beziehung SSS und SetCover

Lemma. $\mathrm{OPT}_{\mathrm{SC}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

Beweis.

Jeder String $s_i \in U$ ist in einem Teilstring π_j enthalten

$\{S(\pi_1), \ldots, S(\pi_k)\}$ ist Lösung für SetCover Instanz mit Kosten $\sum_i |\pi_i|$

Teilstrings π_j, π_{j+2} sind überschneidungsfrei

Jedes Auftreten eines Zeichens liegt in höchstens **zwei** (aufeinanderfolgenden) Teilstrings π_j und π_{j+1}

$\sum_i |\pi_i| \leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

Algorithmus für SSS

1. Erstelle SetCover Instanz U, $\mathcal{S}$, c
2. Berechne ein Set Cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ mit Algorithmus GreedySetCover
3. Gebe $\pi_1 \circ \cdots \circ \pi_k$ als Superstring zurück

Satz. Obiger Algorithmus ist ein Faktor-$2\mathcal{H}_n$-Approximationsalgorithmus für ShortestSuperString.

besser?

Der beste bekannte Approximationsfaktor für ShortestSuperString ist $\frac{71}{30} \approx 2{,}367$

ShortestSuperString lässt sich nicht mit Faktor $\frac{333}{332} \approx 1{,}003$ approximieren (außer P=NP)

Nächste Vorlesung: Dienstag 29.10.2019!