Approximation Algorithms

Lecture 2:
SetCover and ShortestSuperString

Part I:
SetCover

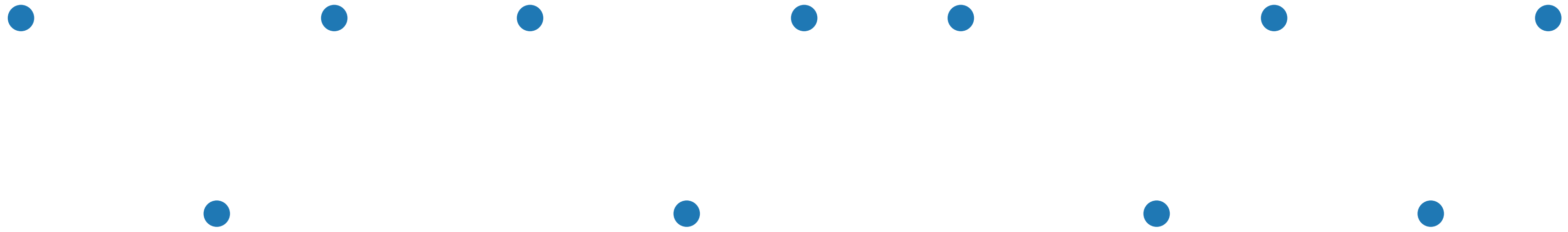
Philipp Kindermann Summer Semester 2020

SetCover (card.)

Given a **ground set** U

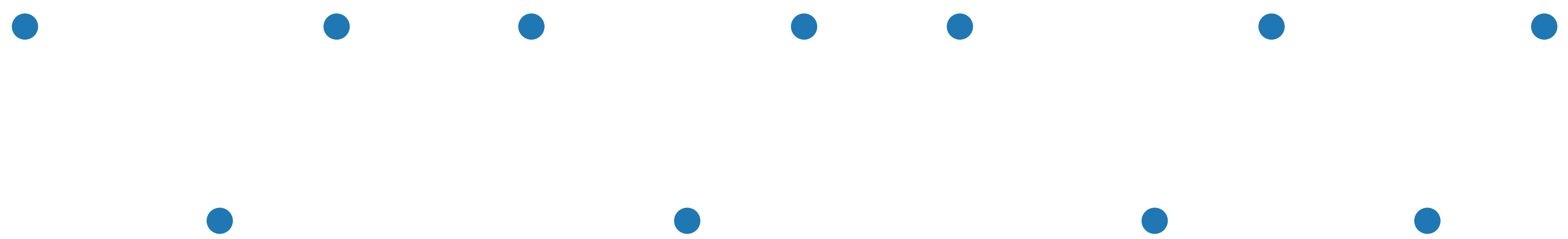
SetCover (card.)

Given a **ground set** U



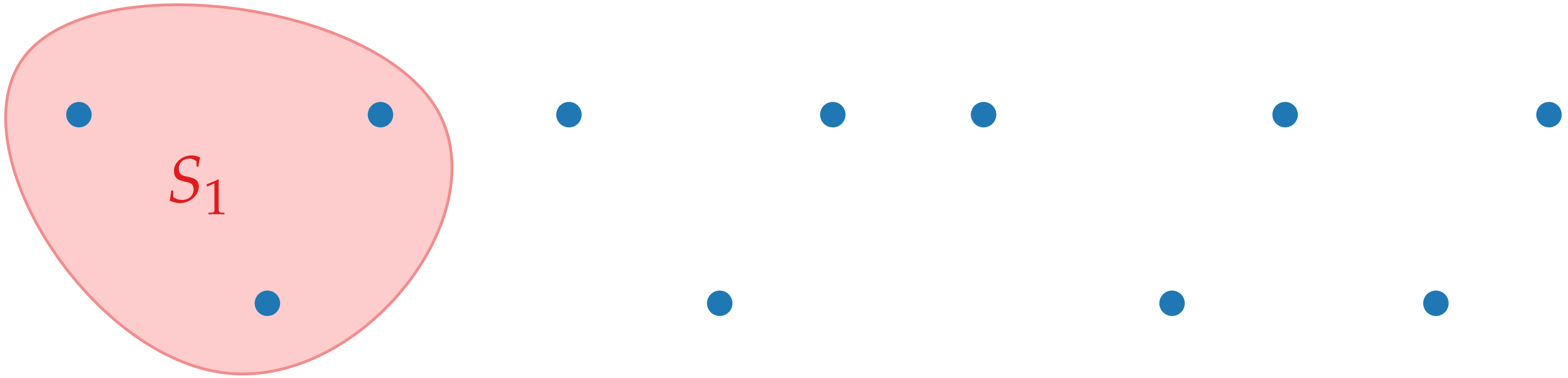
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U



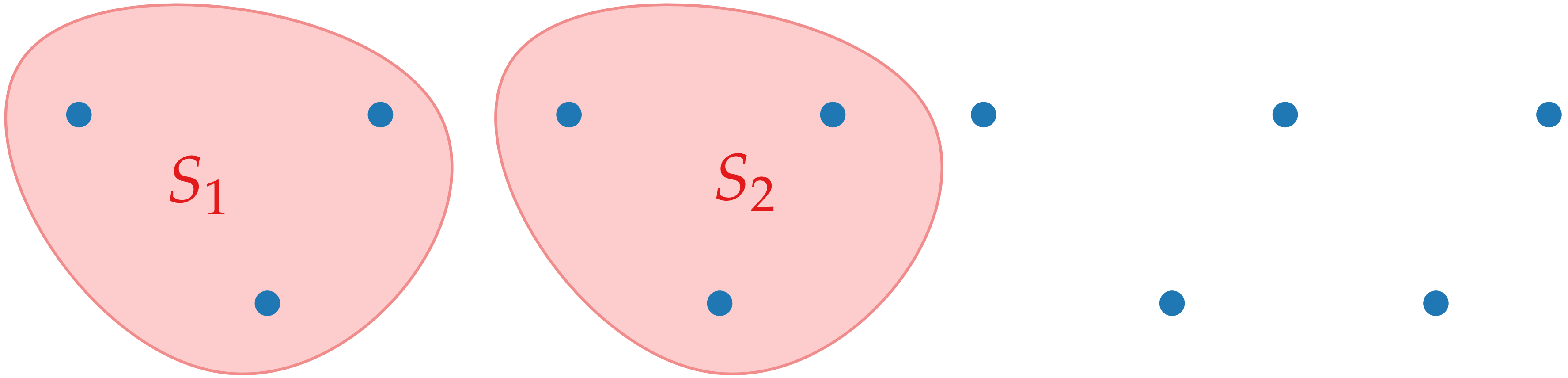
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U



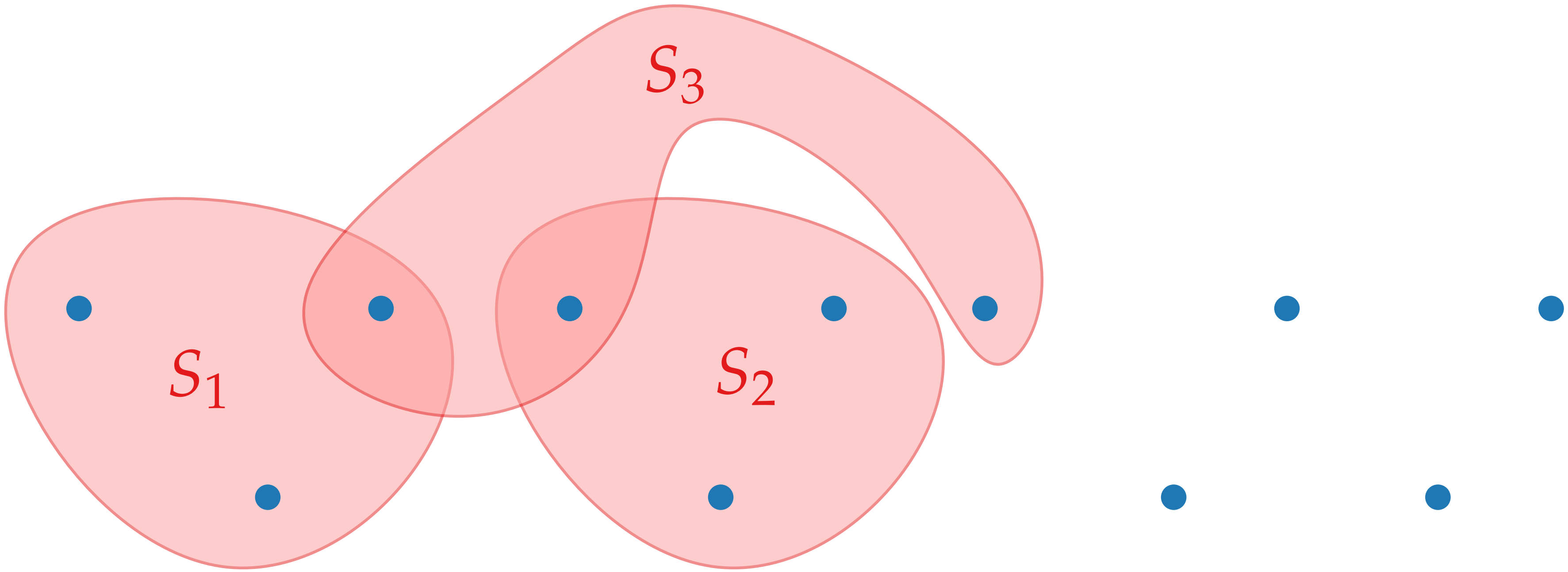
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U



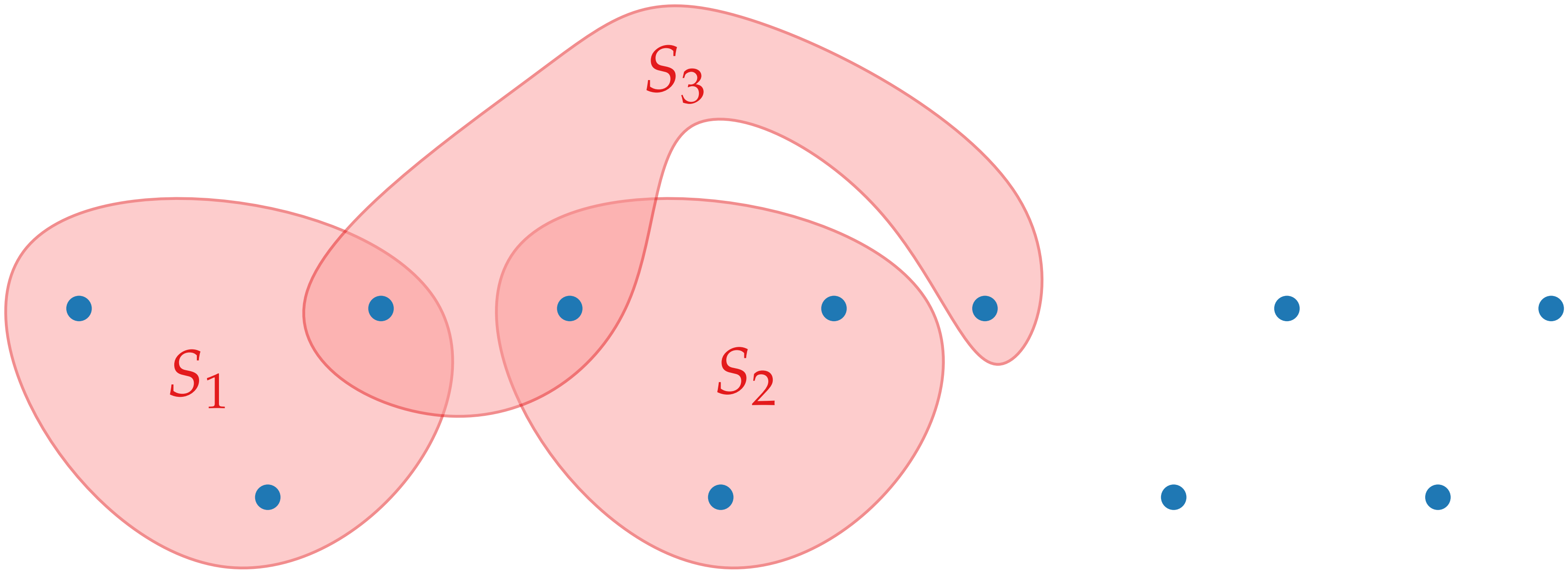
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U



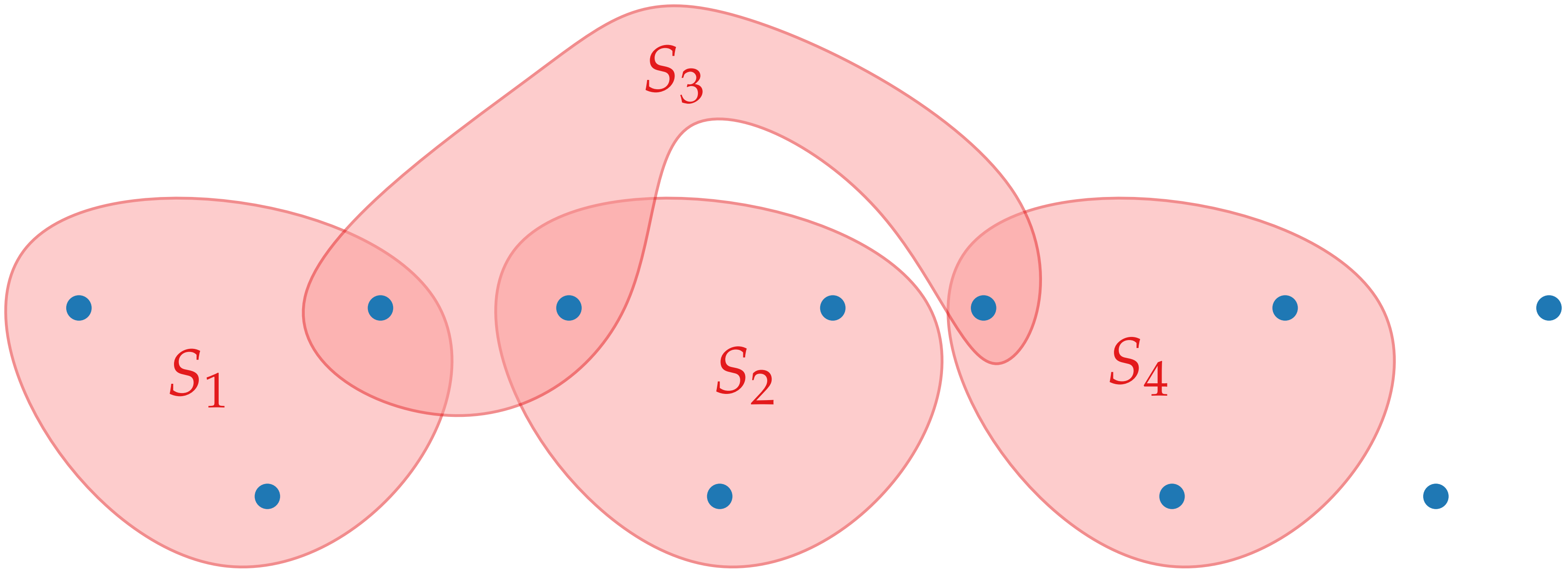
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.



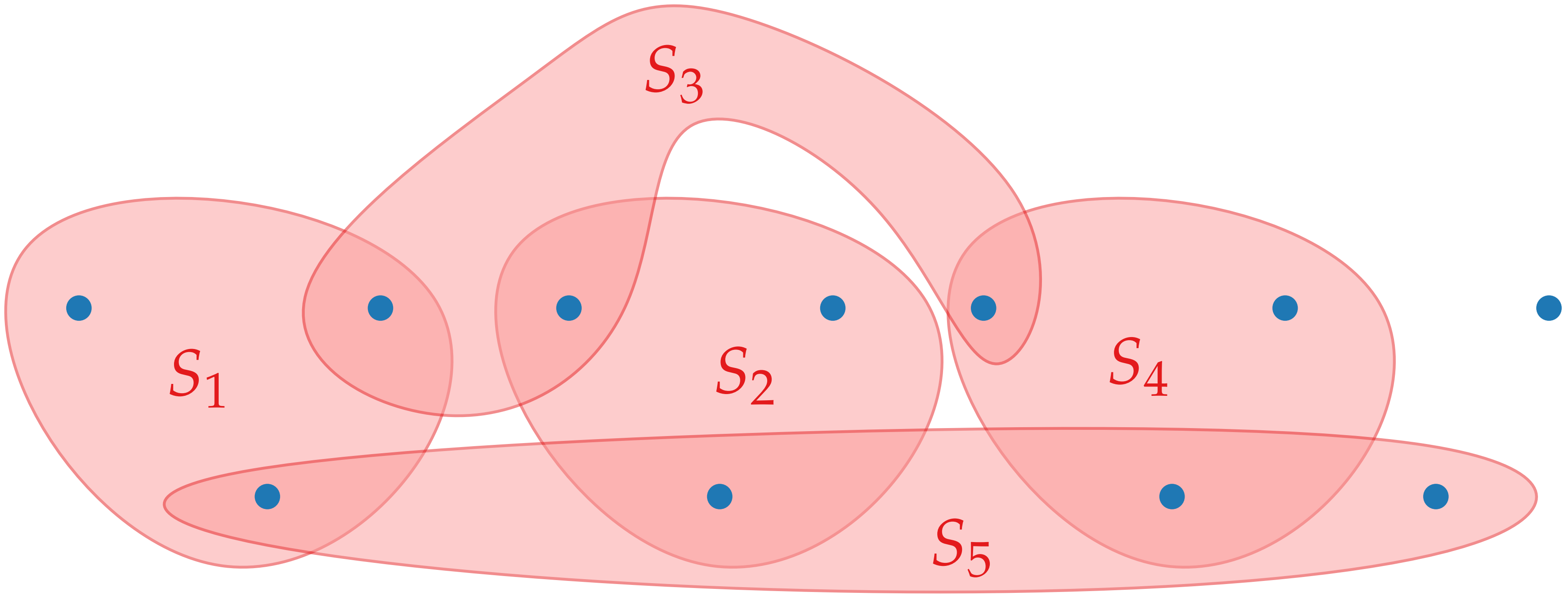
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.



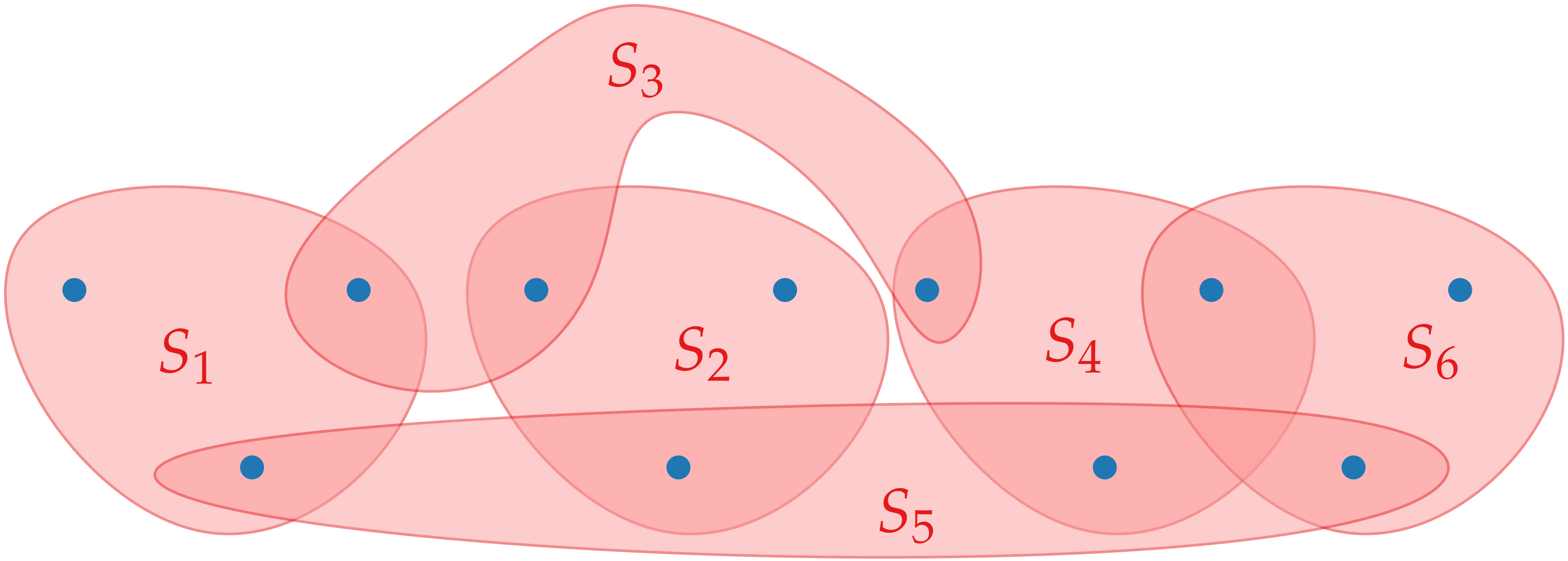
SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.



SetCover (card.)

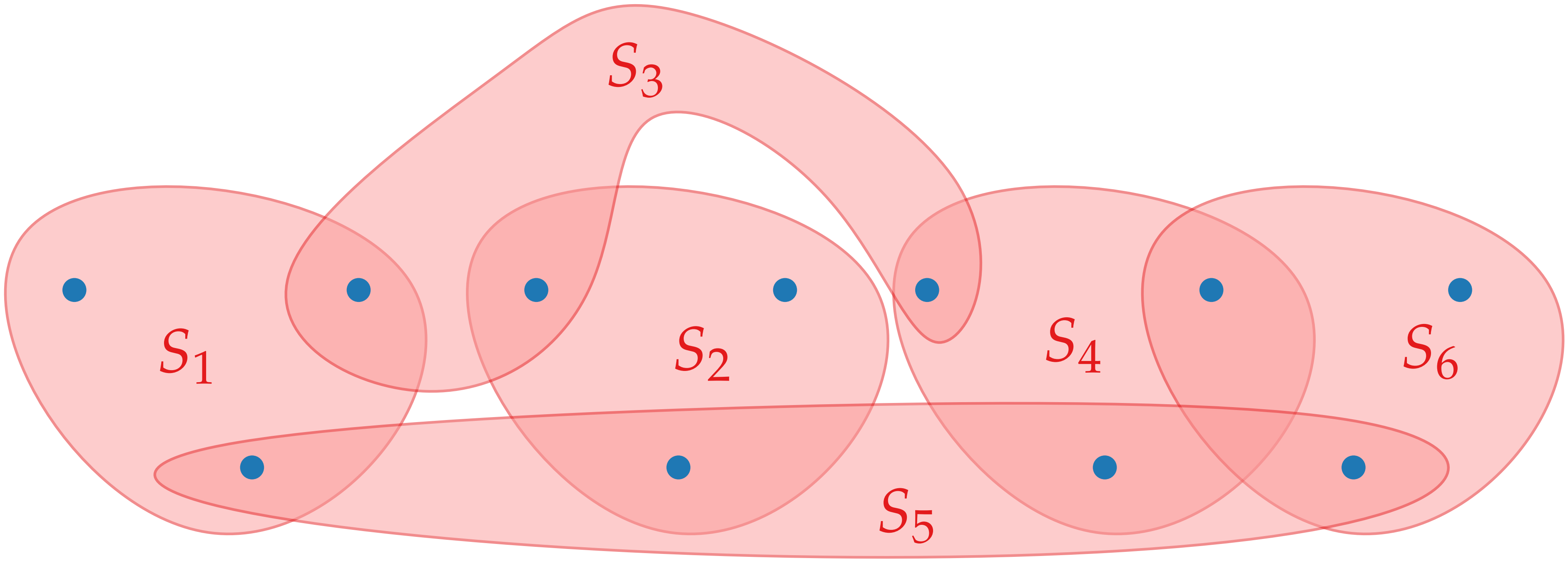
Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.



SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

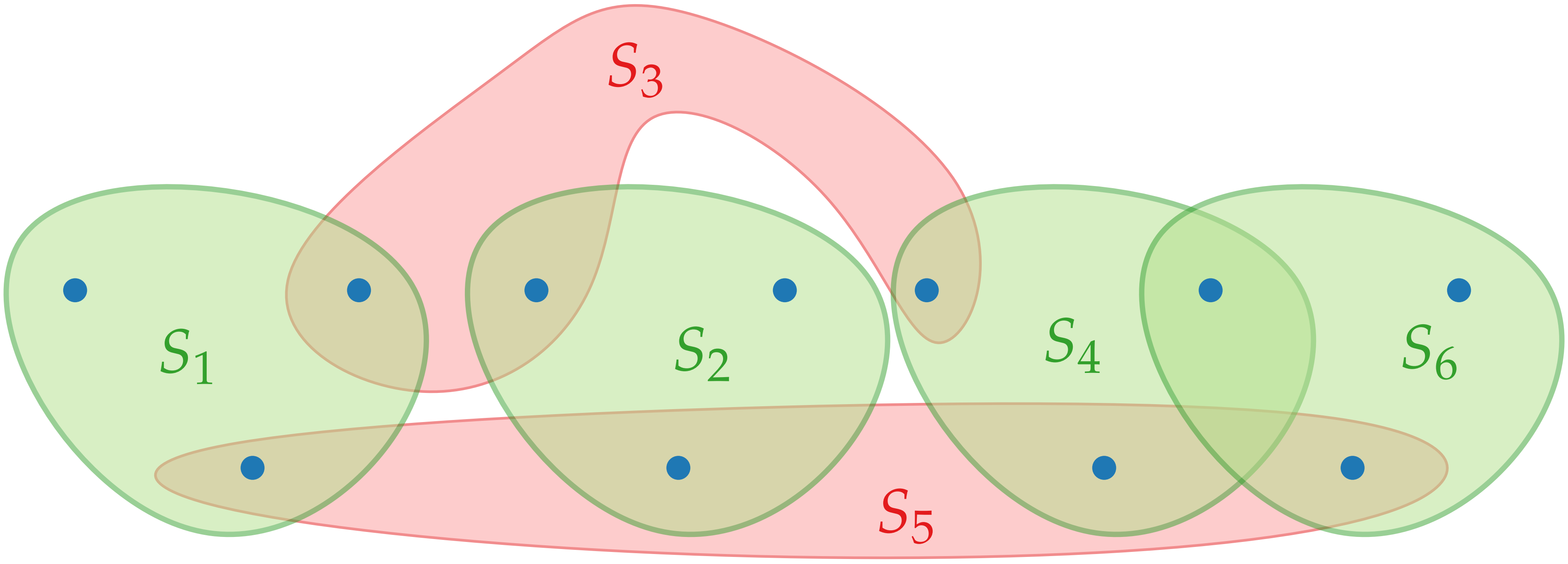
Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal cardinality.



SetCover (card.)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

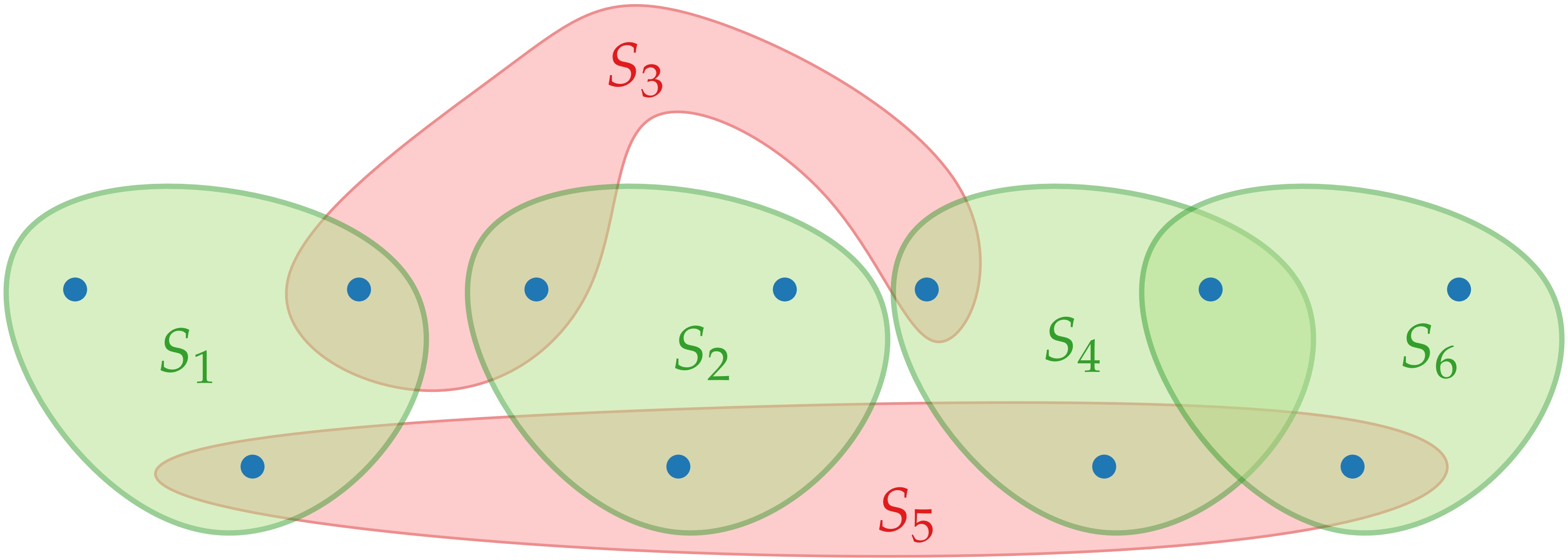
Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal cardinality.



SetCover (general)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal cardinality.

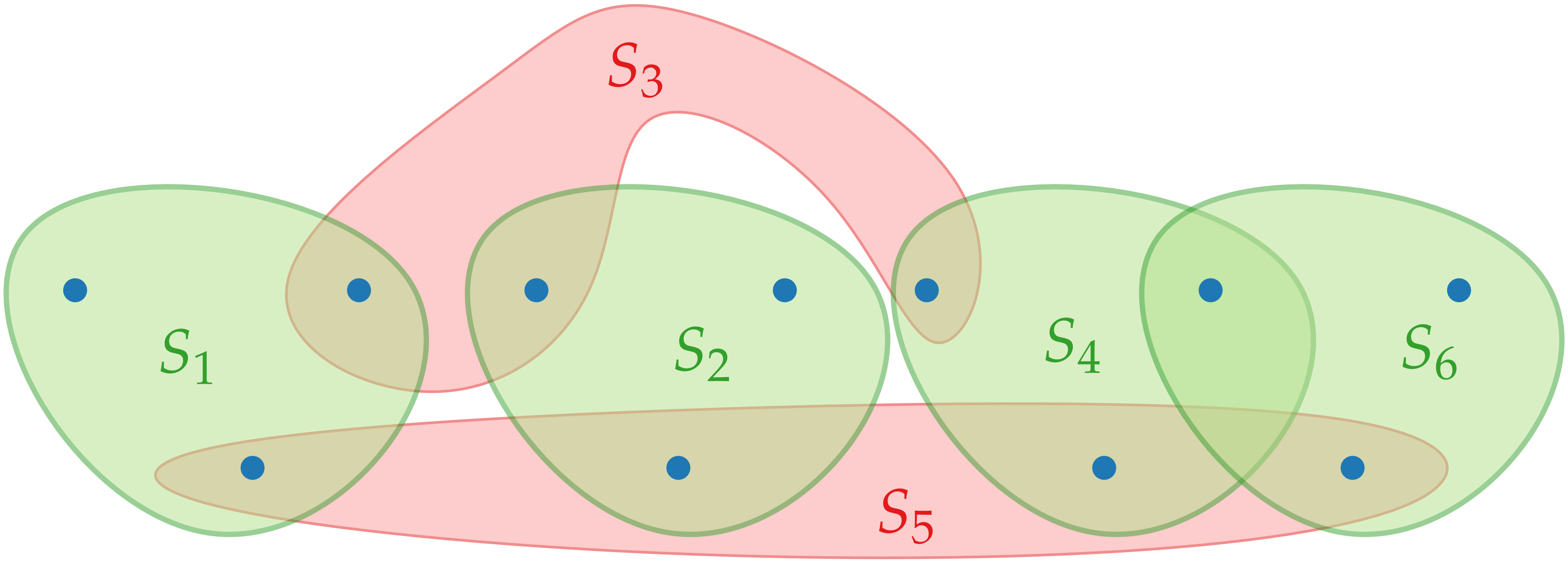


SetCover (general)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

Each $S \in \mathcal{S}$ has cost $c(S) > 0$.

Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal cardinality.

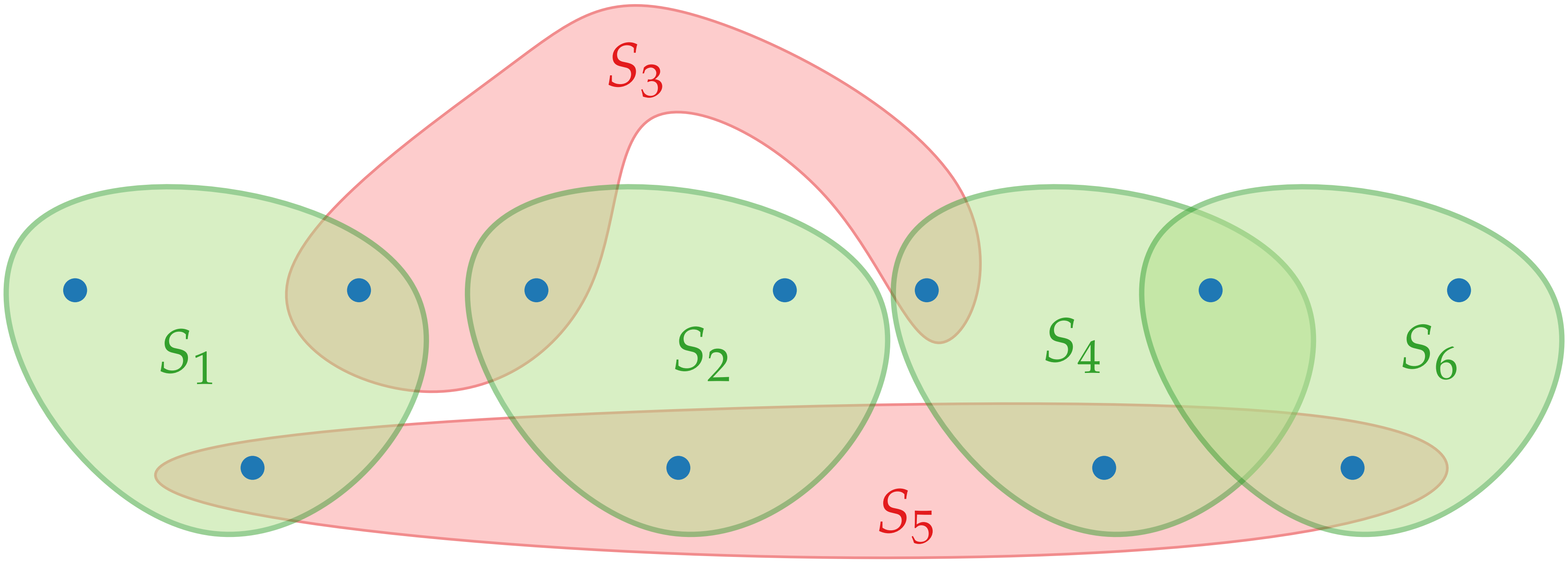


SetCover (general)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

Each $S \in \mathcal{S}$ has cost $c(S) > 0$.

Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal ~~cardinality.~~ total cost $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c(S)$.

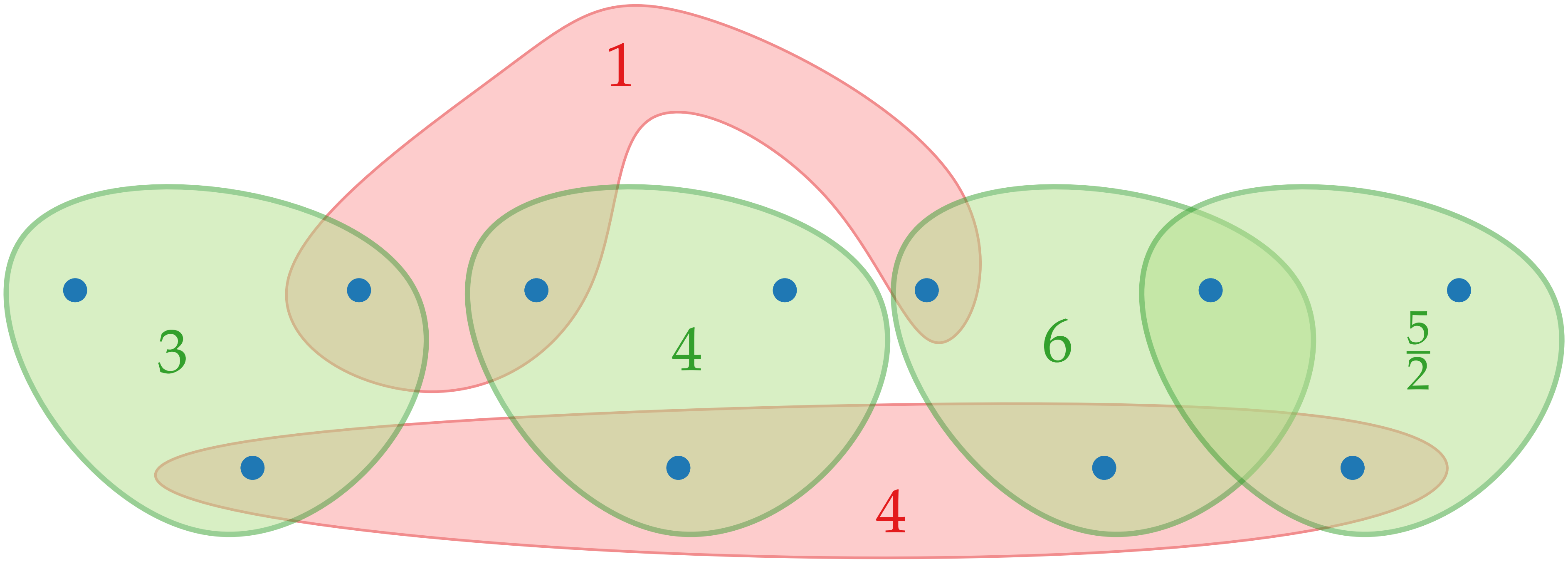


SetCover (general)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

Each $S \in \mathcal{S}$ has cost $c(S) > 0$.

Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal ~~cardinality.~~ total cost $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c(S)$.

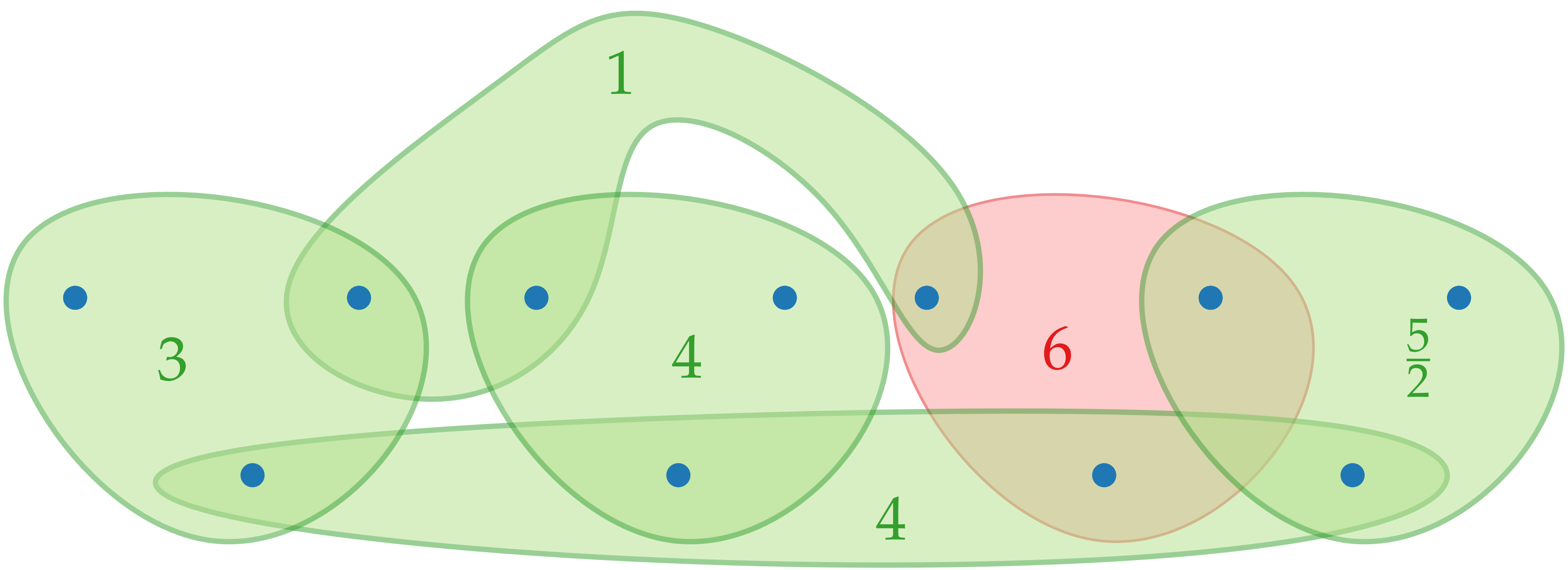


SetCover (general)

Given a **ground set** U and a family $\mathcal{S}$ of **subsets** of U with $\bigcup \mathcal{S} = U$.

Each $S \in \mathcal{S}$ has cost $c(S) > 0$.

Find a **cover** $\mathcal{S}' \subseteq \mathcal{S}$ of U (i.e. with $\bigcup \mathcal{S}' = U$) of minimal ~~cardinality.~~ total cost $c(\mathcal{S}') := \sum_{S \in \mathcal{S}'} c(S)$.



Approximation Algorithms

Lecture 2:
SetCover and ShortestSuperString

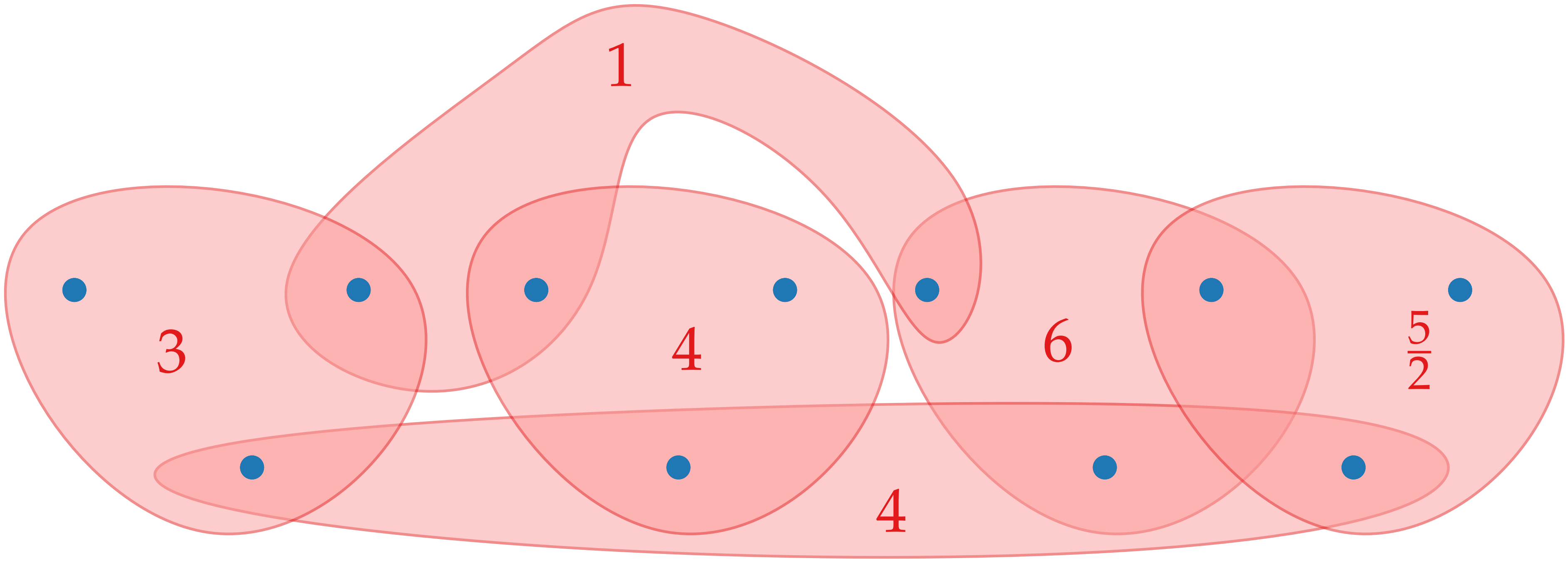
Part II:
Greedy for SetCover

Philipp Kindermann

Summer Semester 2020

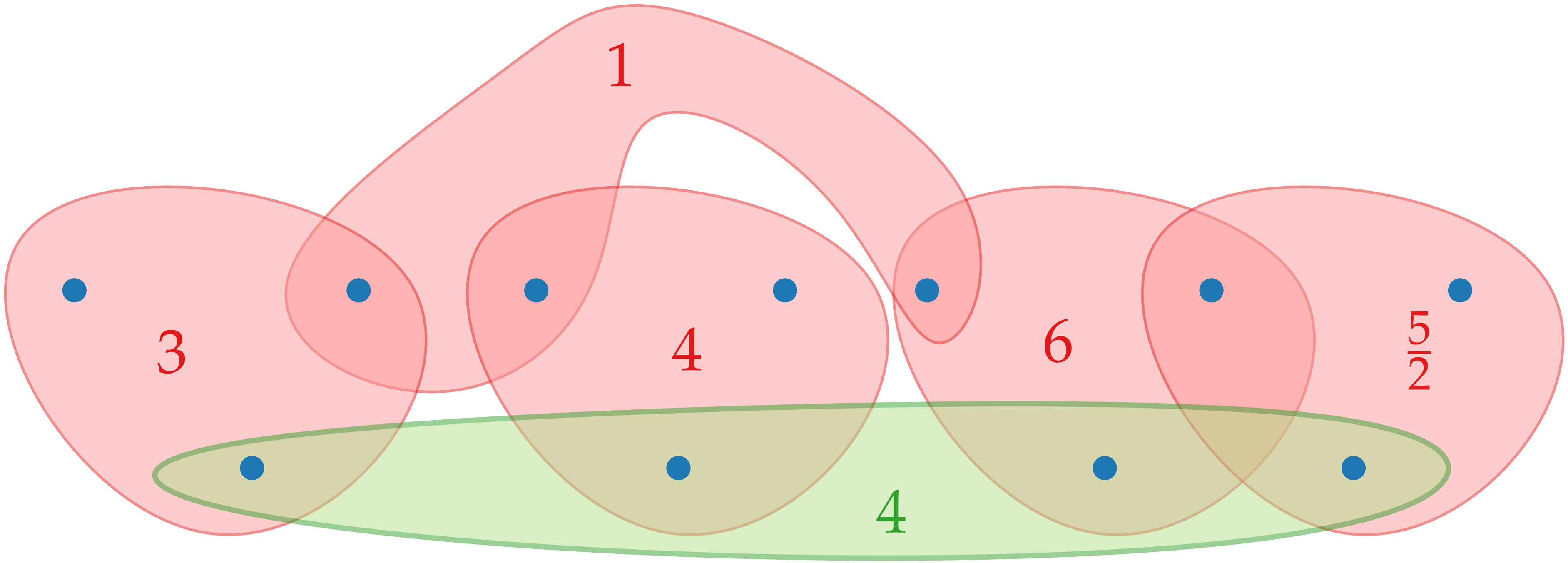
Iterative "Buying" of Elements

What is the real cost of picking a set?



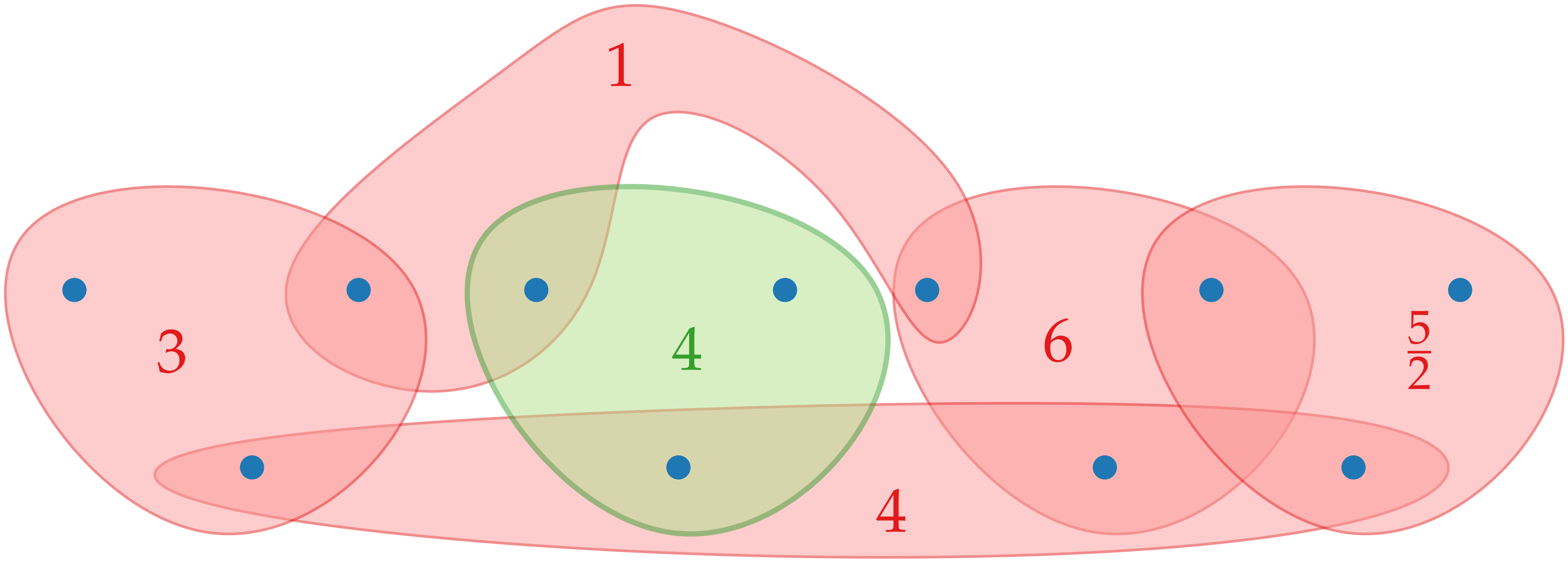
Iterative "Buying" of Elements

What is the real cost of picking a set?



Iterative "Buying" of Elements

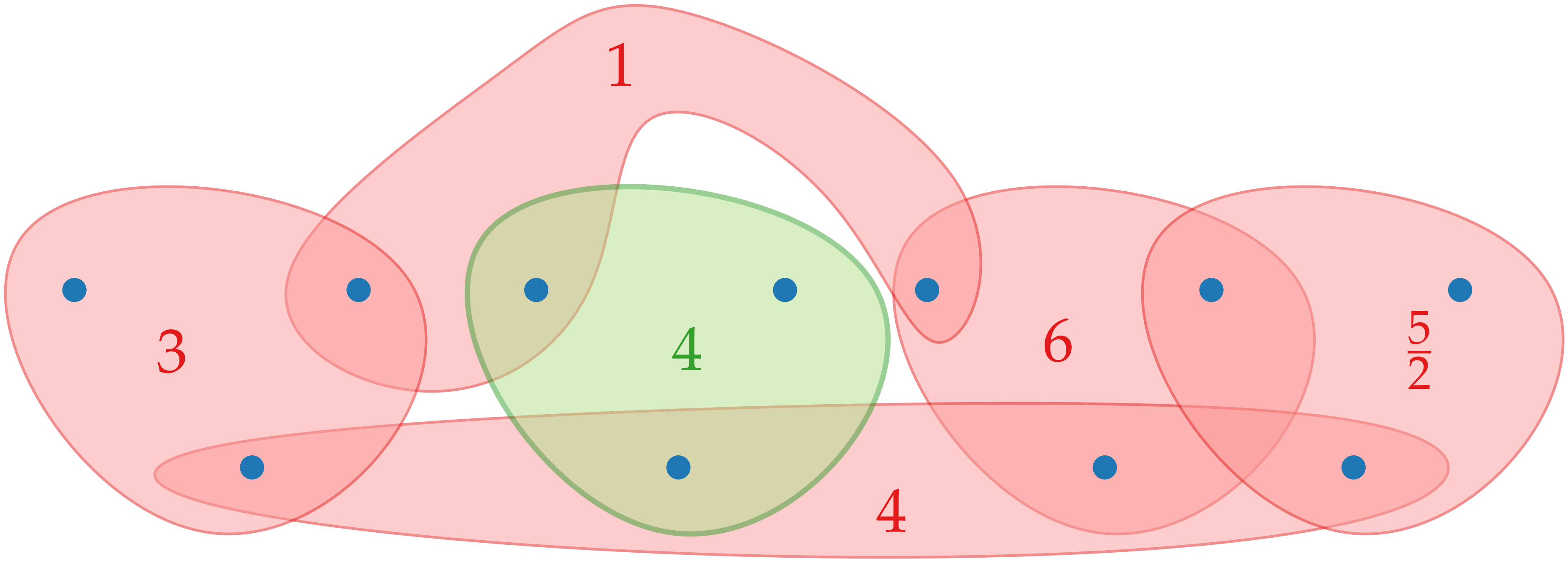
What is the real cost of picking a set?



Iterative "Buying" of Elements

What is the real cost of picking a set?

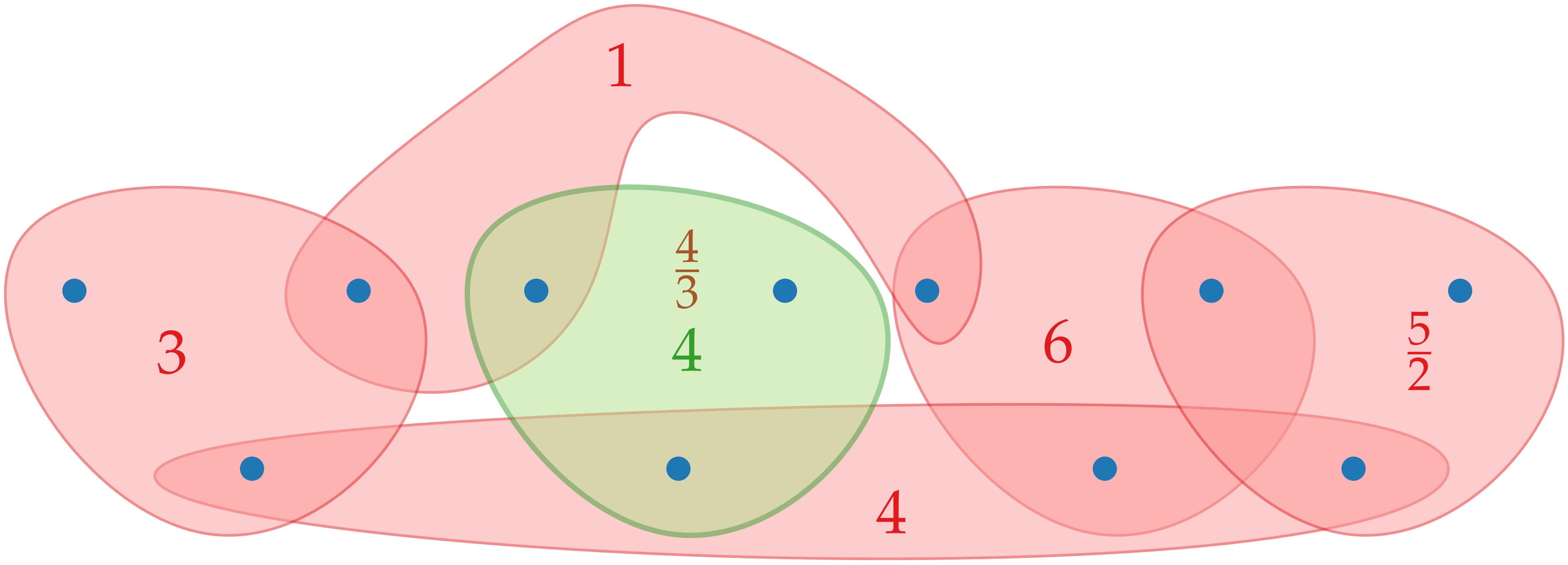
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative "Buying" of Elements

What is the real cost of picking a set?

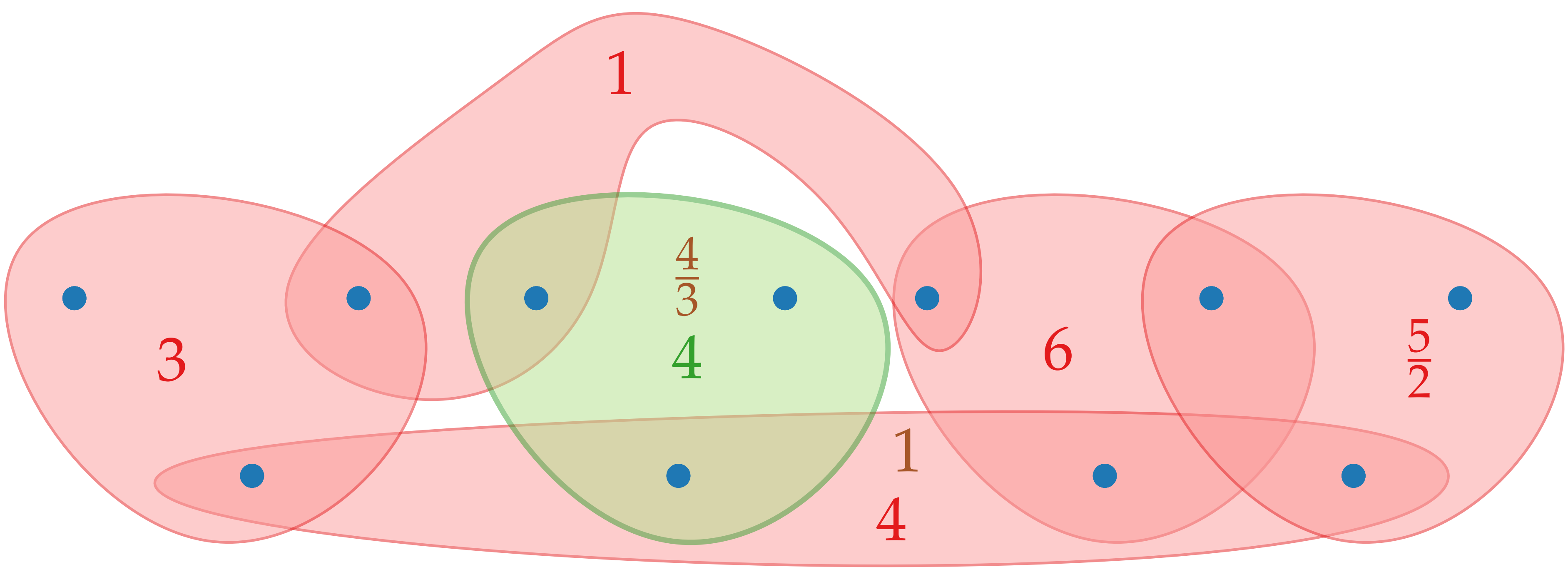
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative "Buying" of Elements

What is the real cost of picking a set?

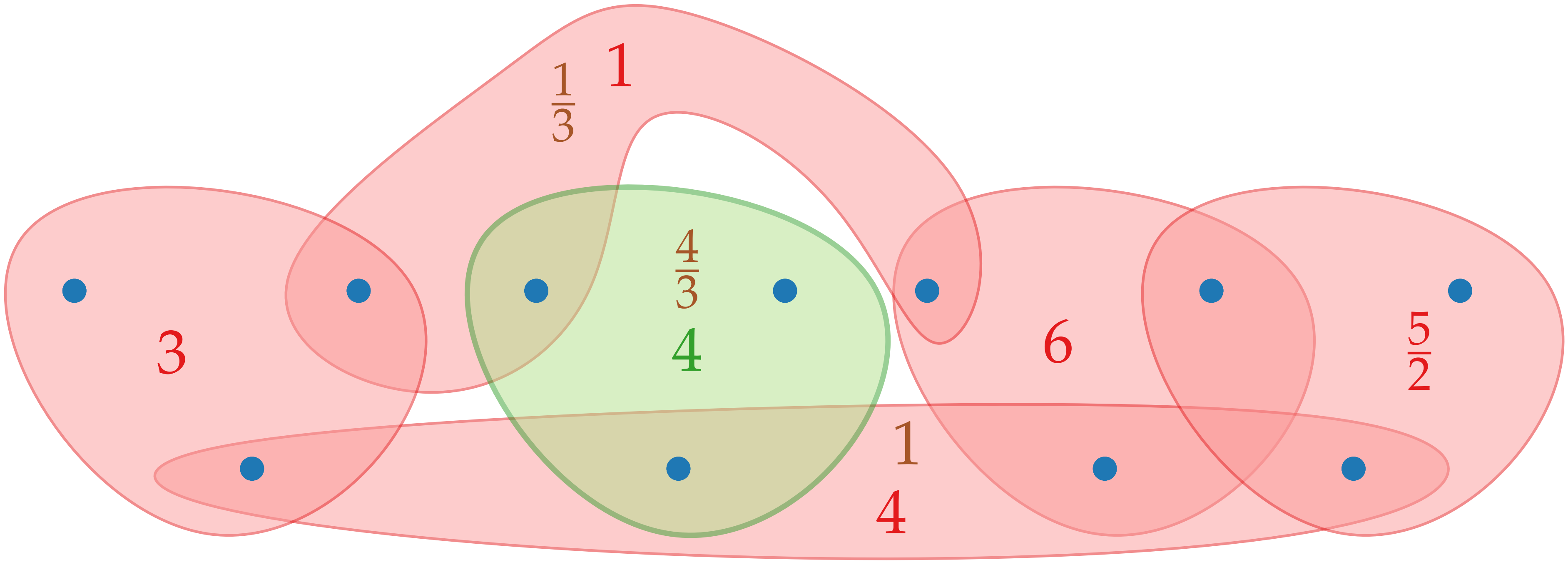
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative "Buying" of Elements

What is the real cost of picking a set?

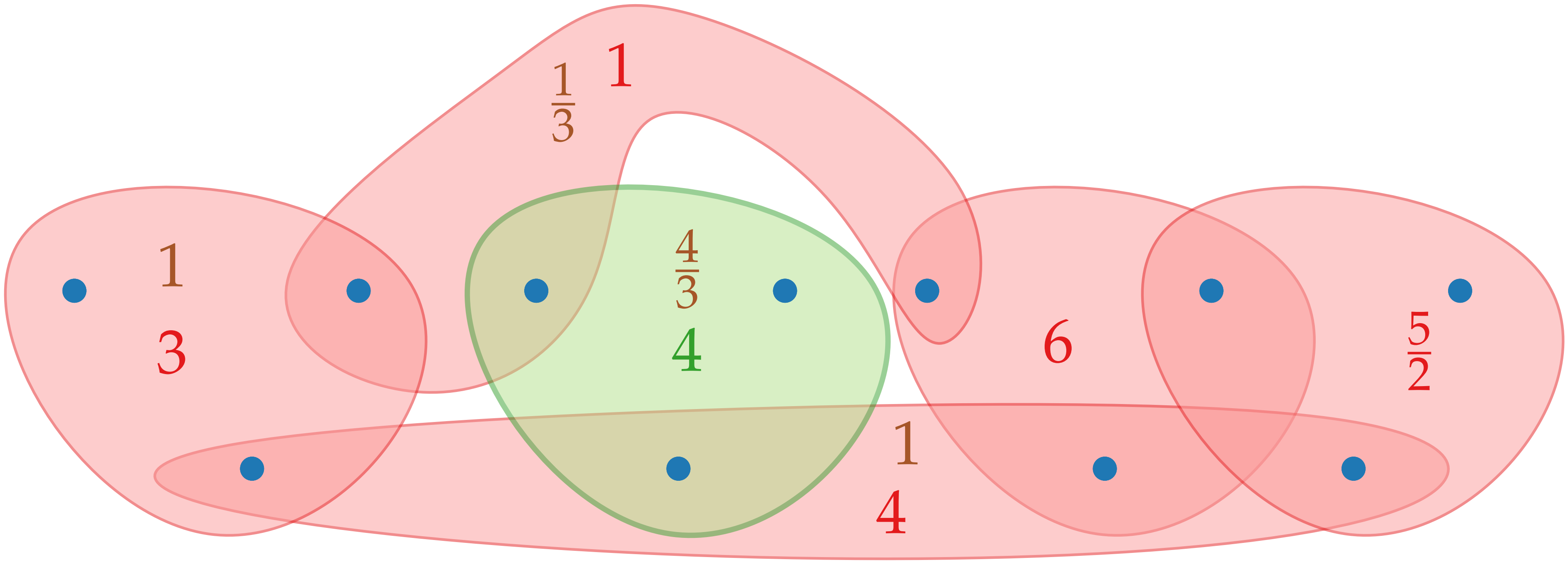
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative "Buying" of Elements

What is the real cost of picking a set?

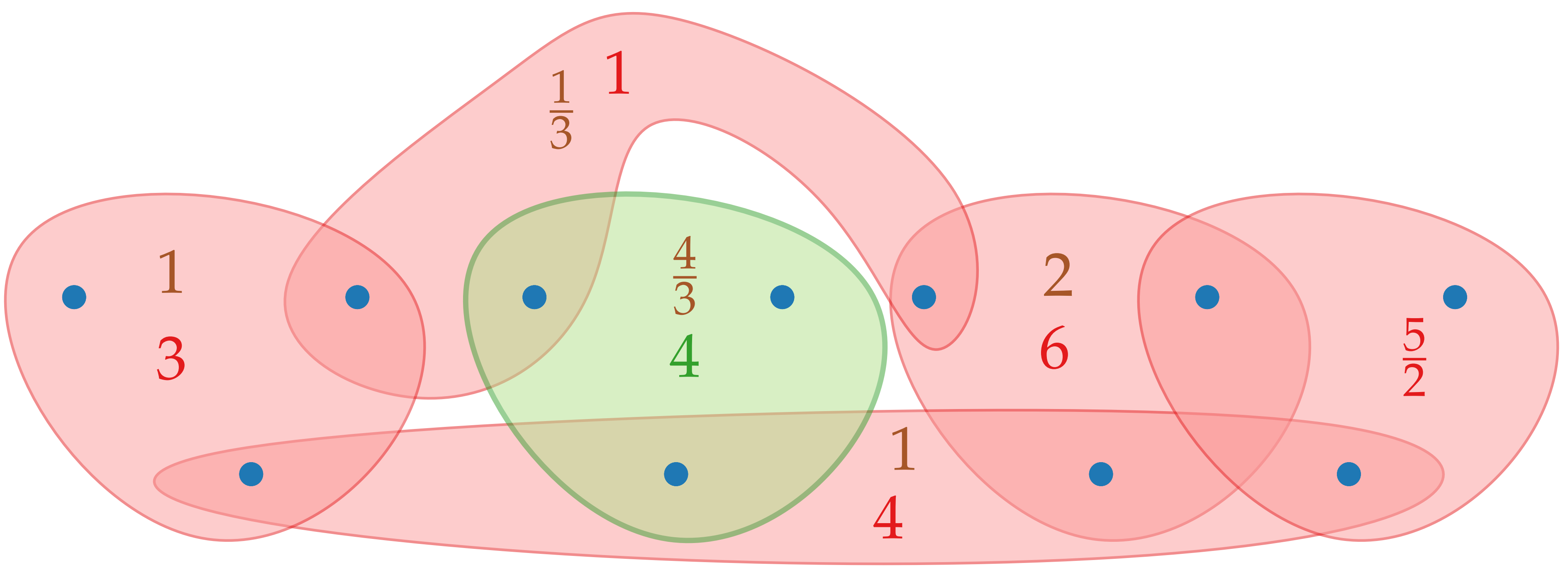
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative "Buying" of Elements

What is the real cost of picking a set?

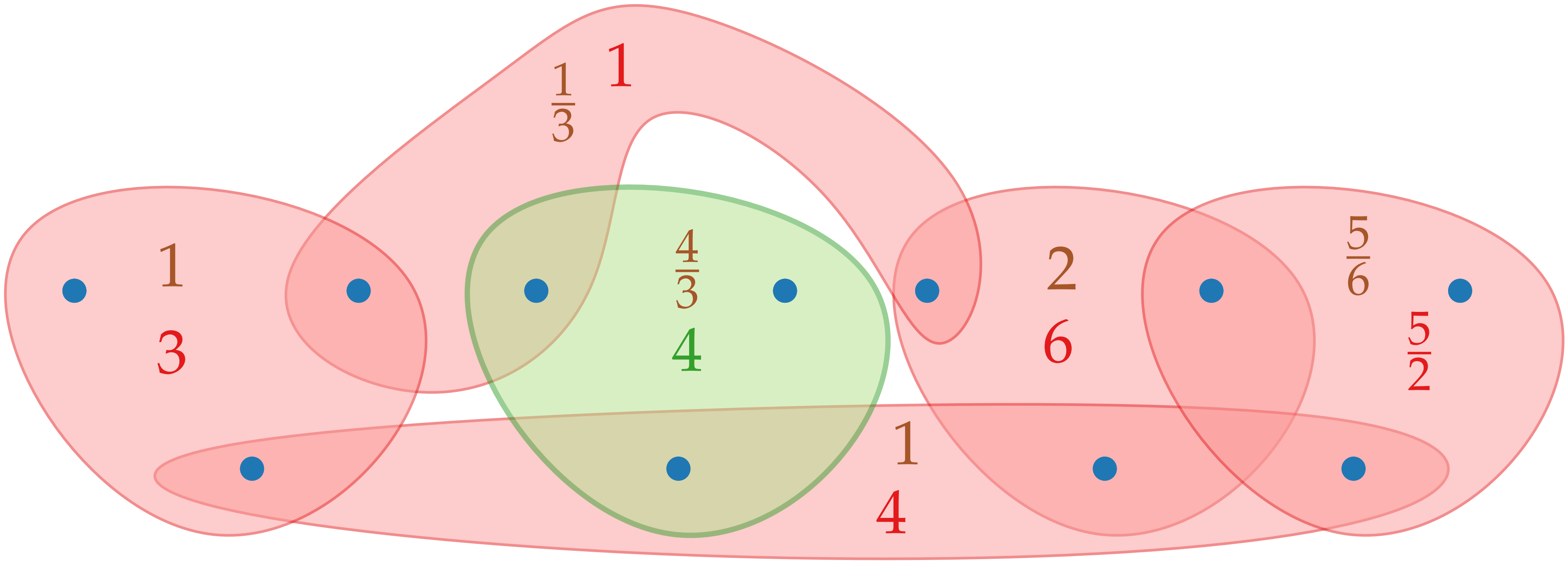
Set with k elements and cost c has unit cost $\frac{c}{k}$.



Iterative “Buying” of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

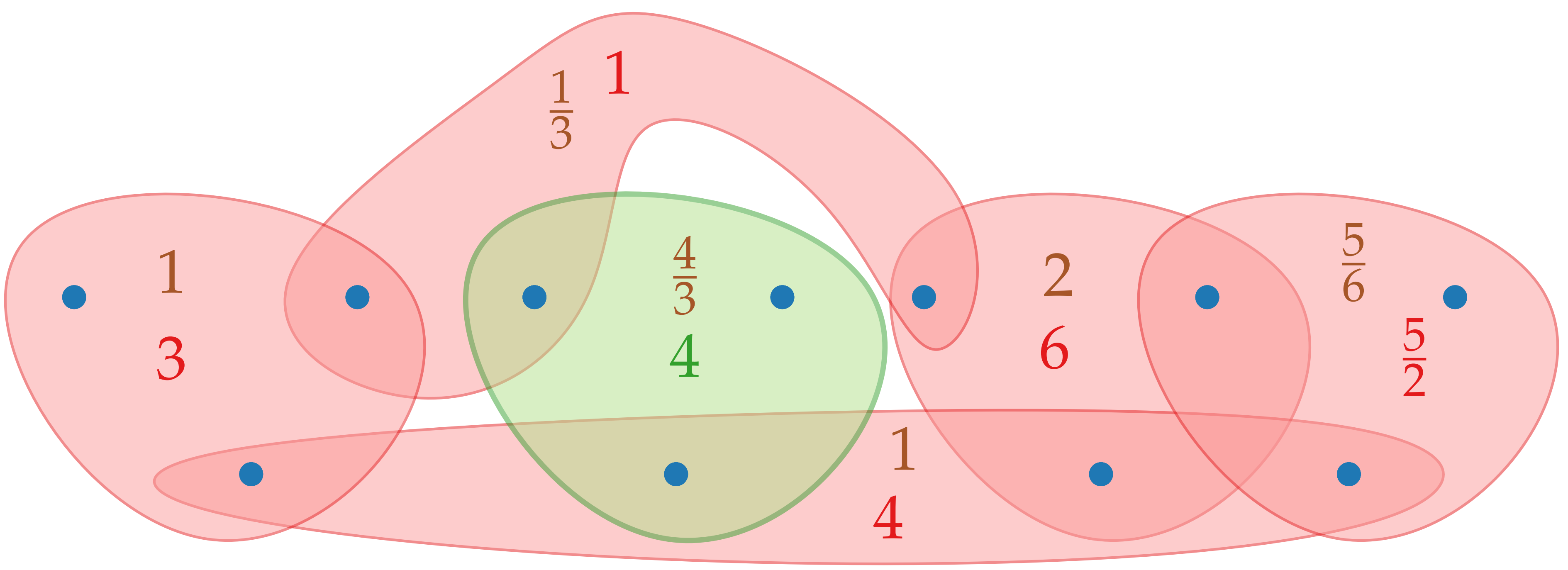


Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?



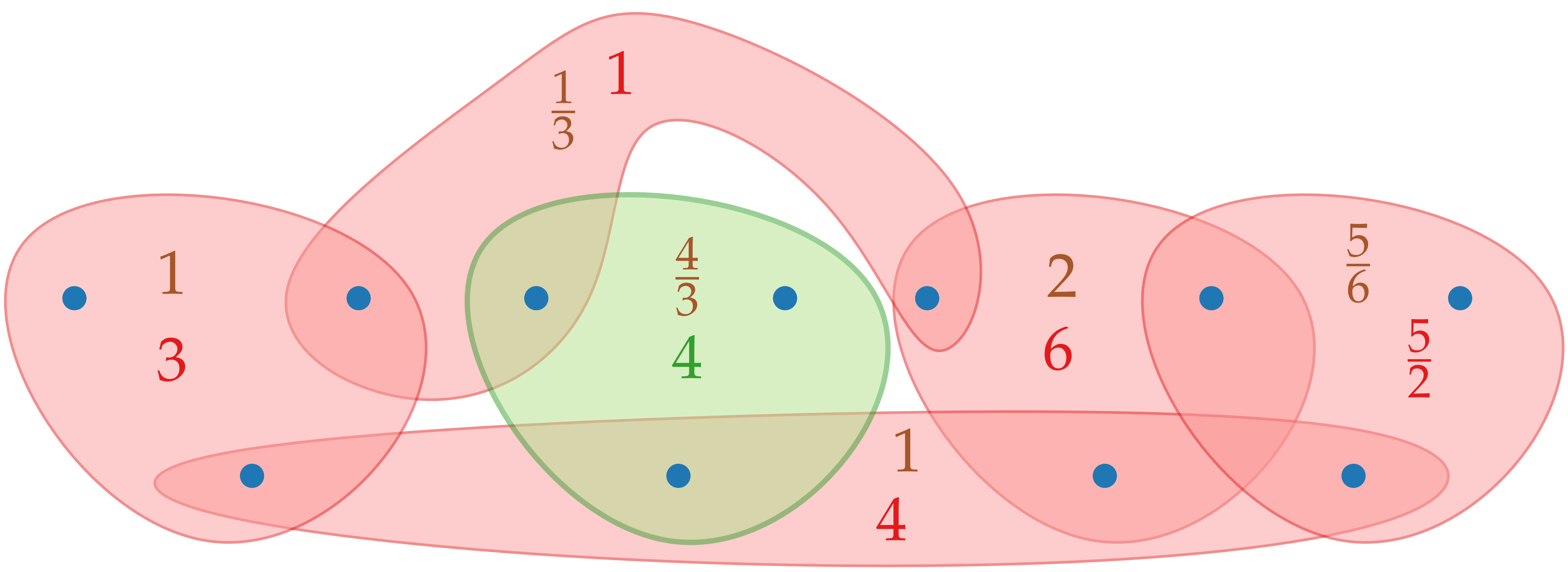
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



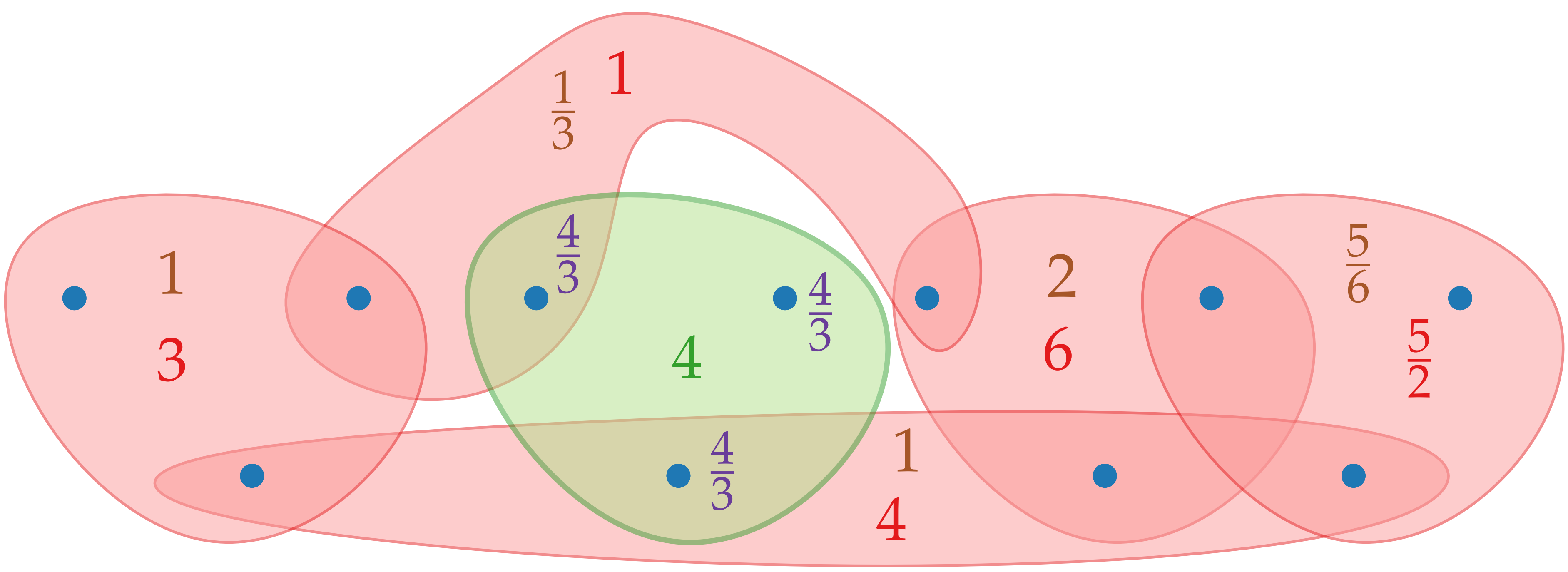
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



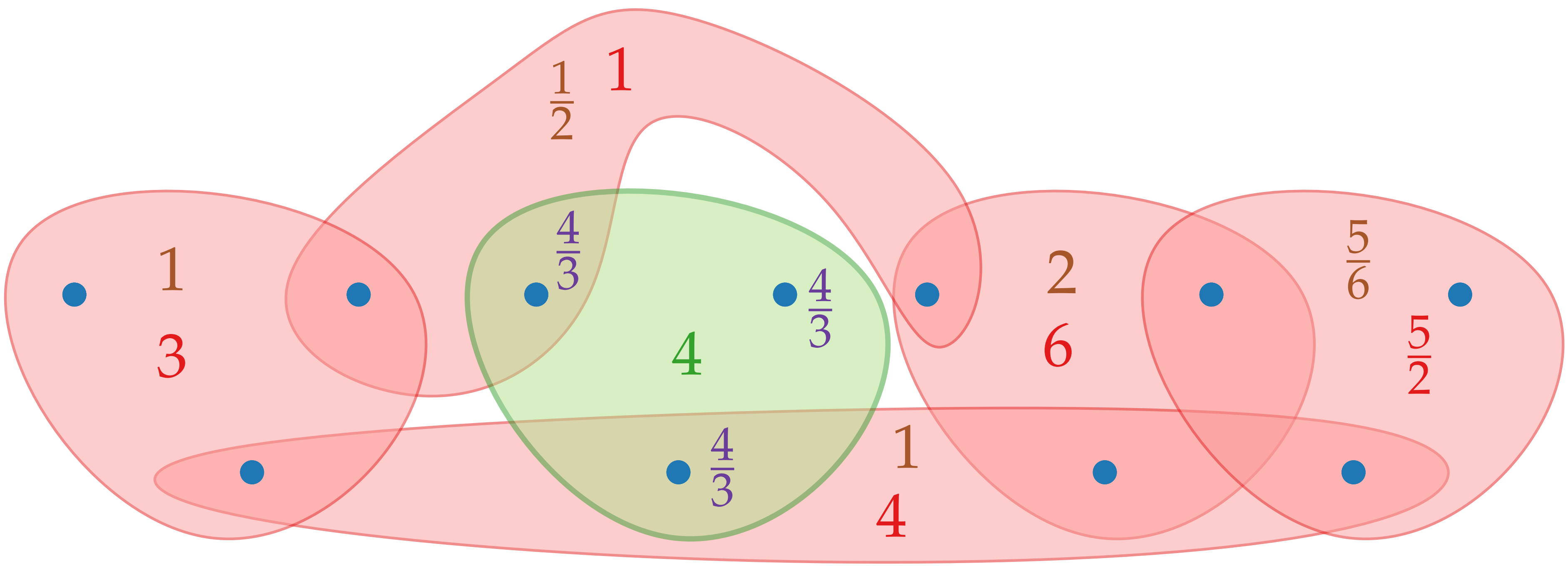
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



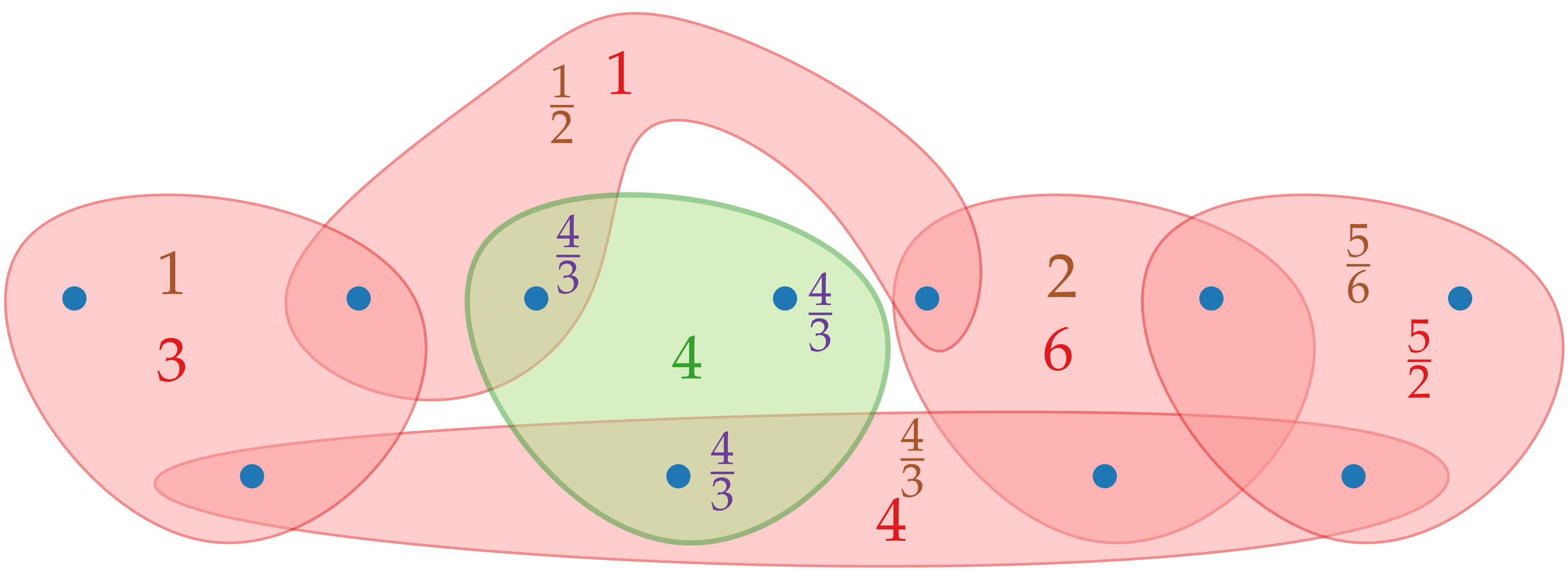
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

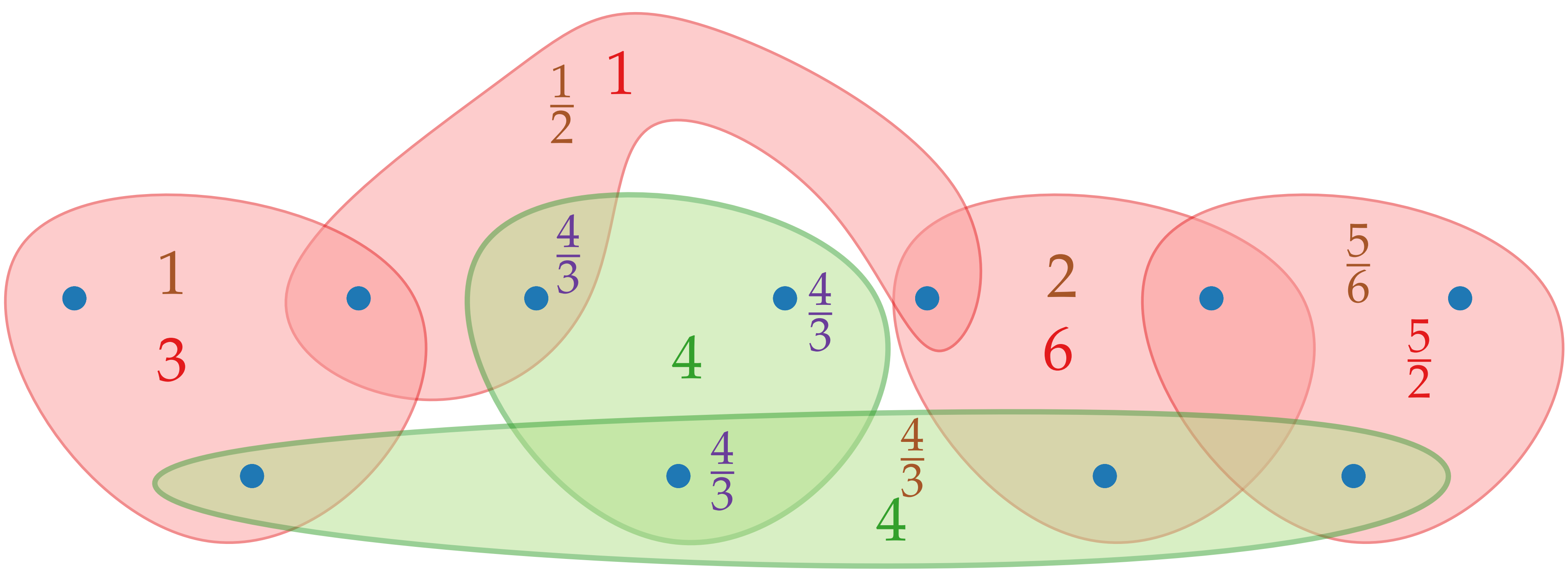
What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



Iterative "Buying" of Elements

What is the real cost of picking a set?
Set with k elements and cost c has unit cost $\frac{c}{k}$.
What happens if we "buy" a set?
Fix price of elements bought and recompute unit cost.



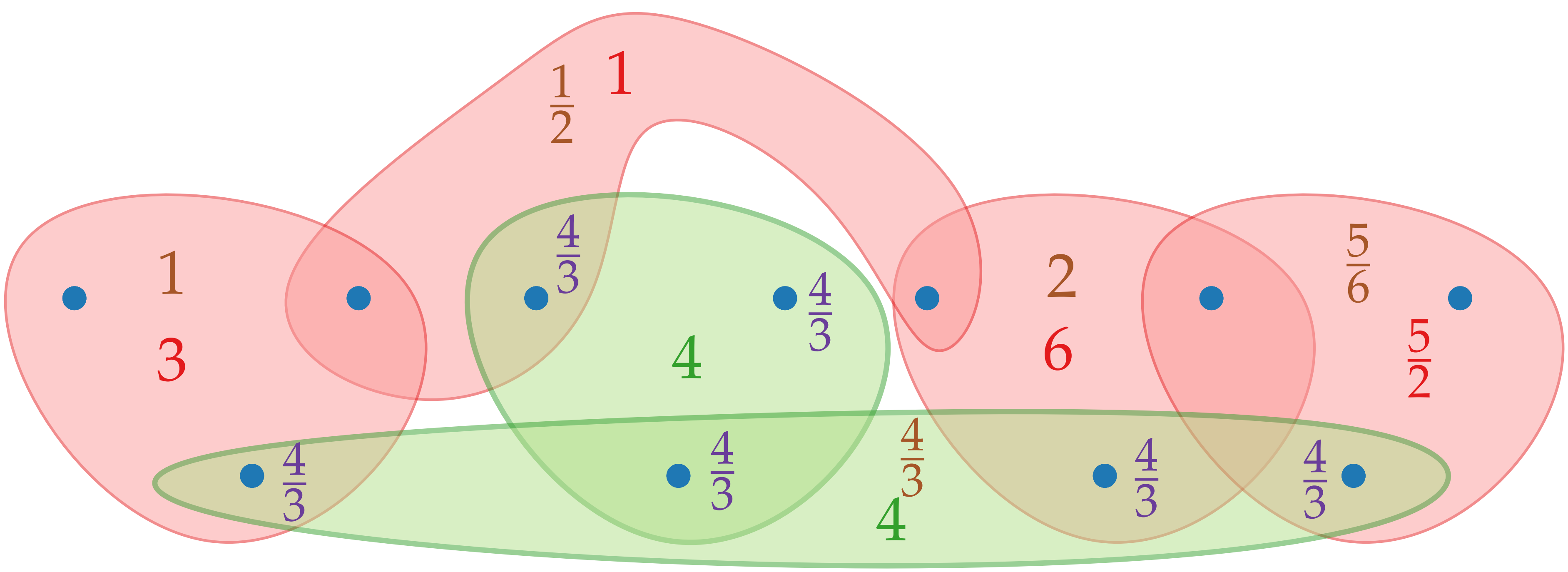
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



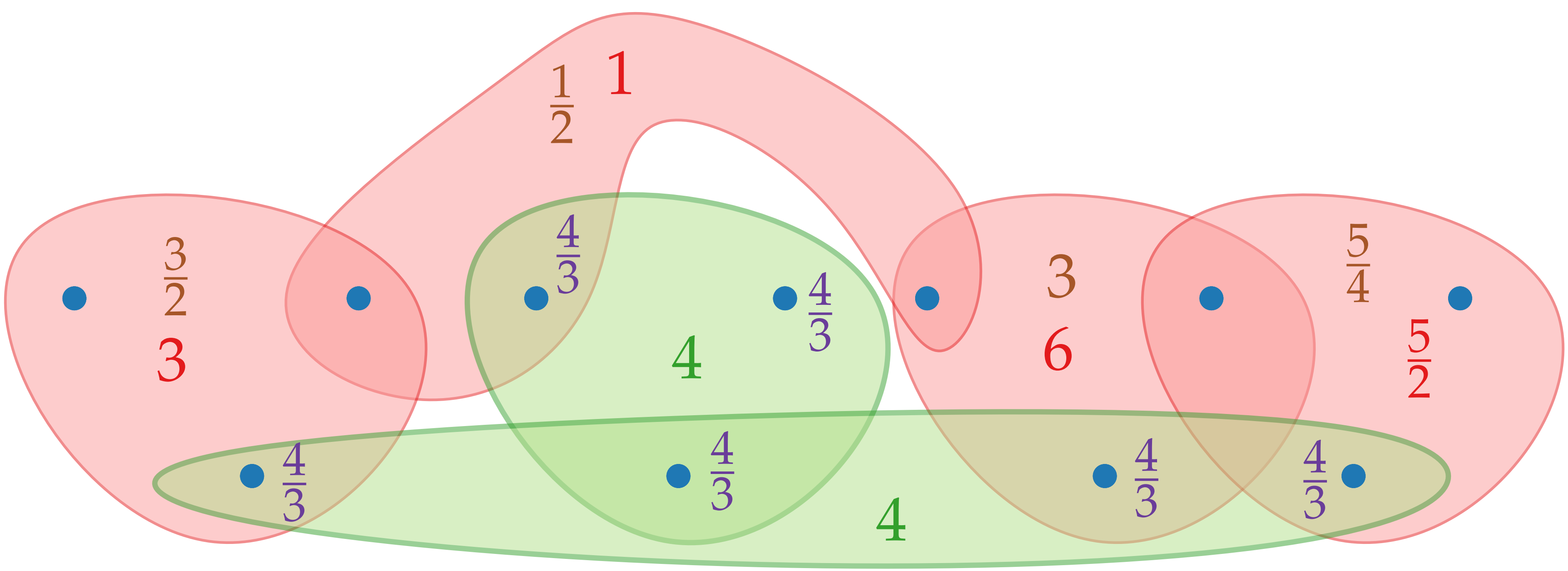
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



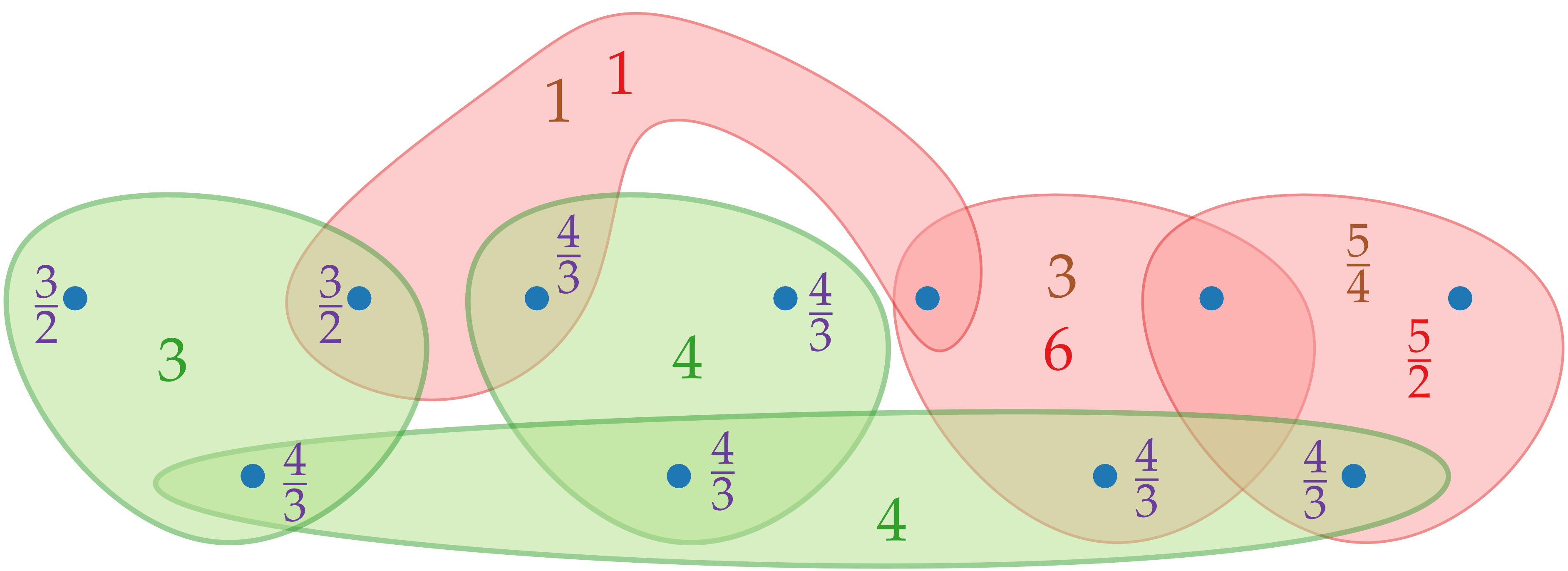
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



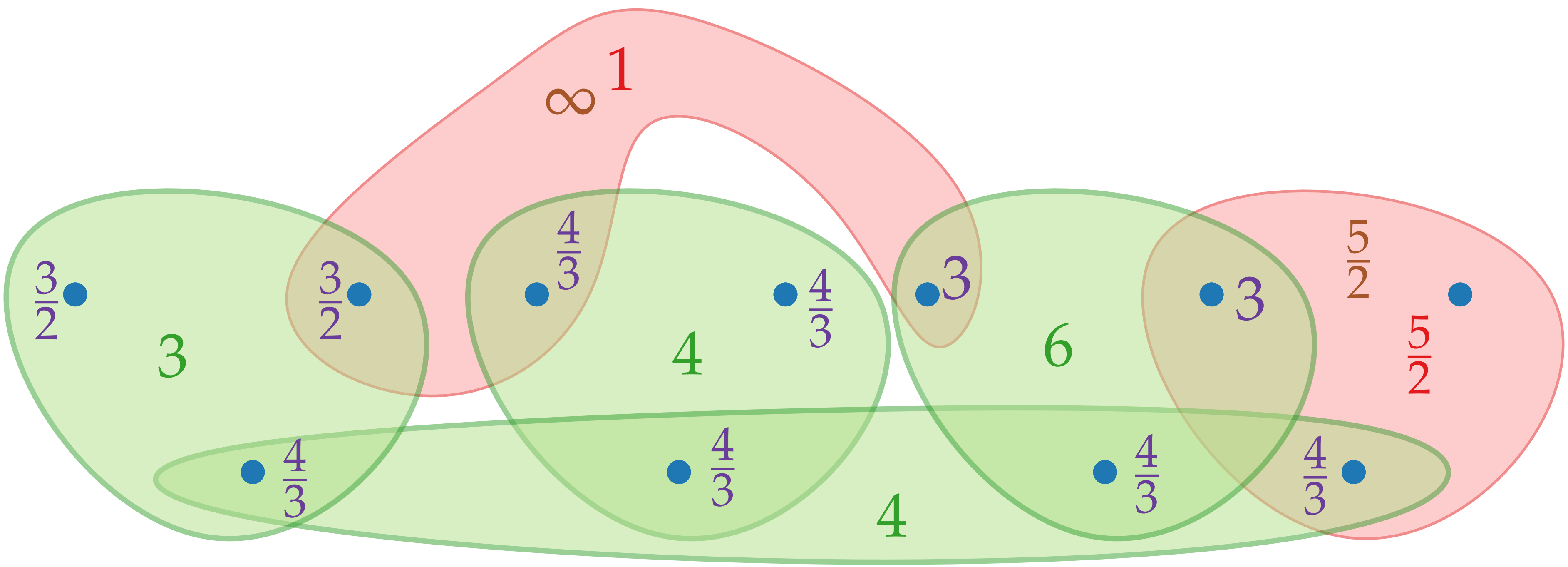
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



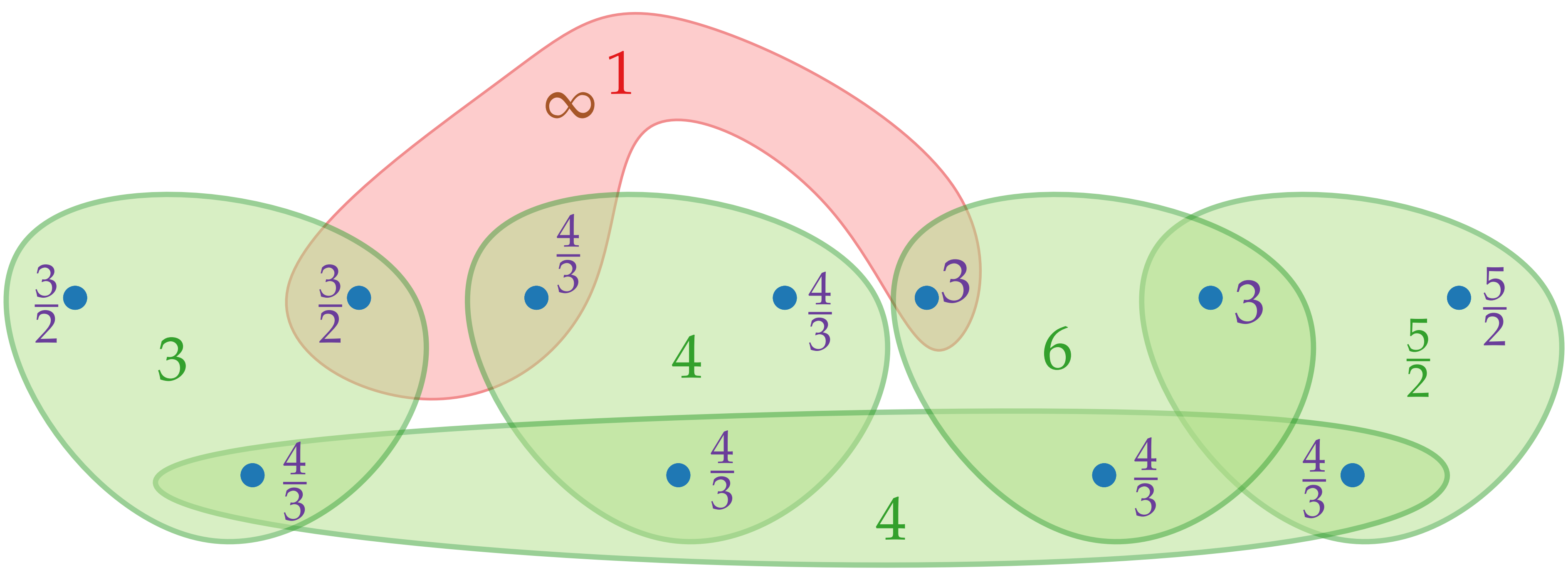
Iterative "Buying" of Elements

What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.



Iterative "Buying" of Elements

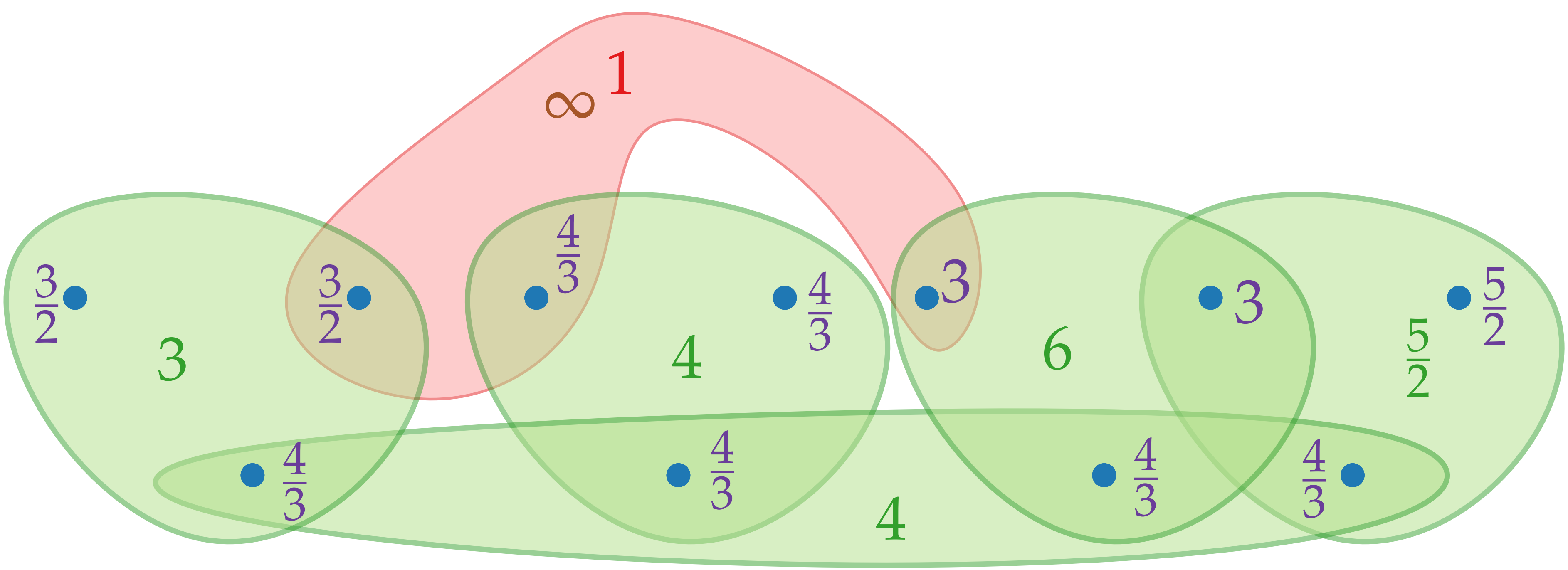
What is the real cost of picking a set?

Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.

total cost: $\sum_{u \in U} \text{price}(u)$



Iterative "Buying" of Elements

What is the real cost of picking a set?

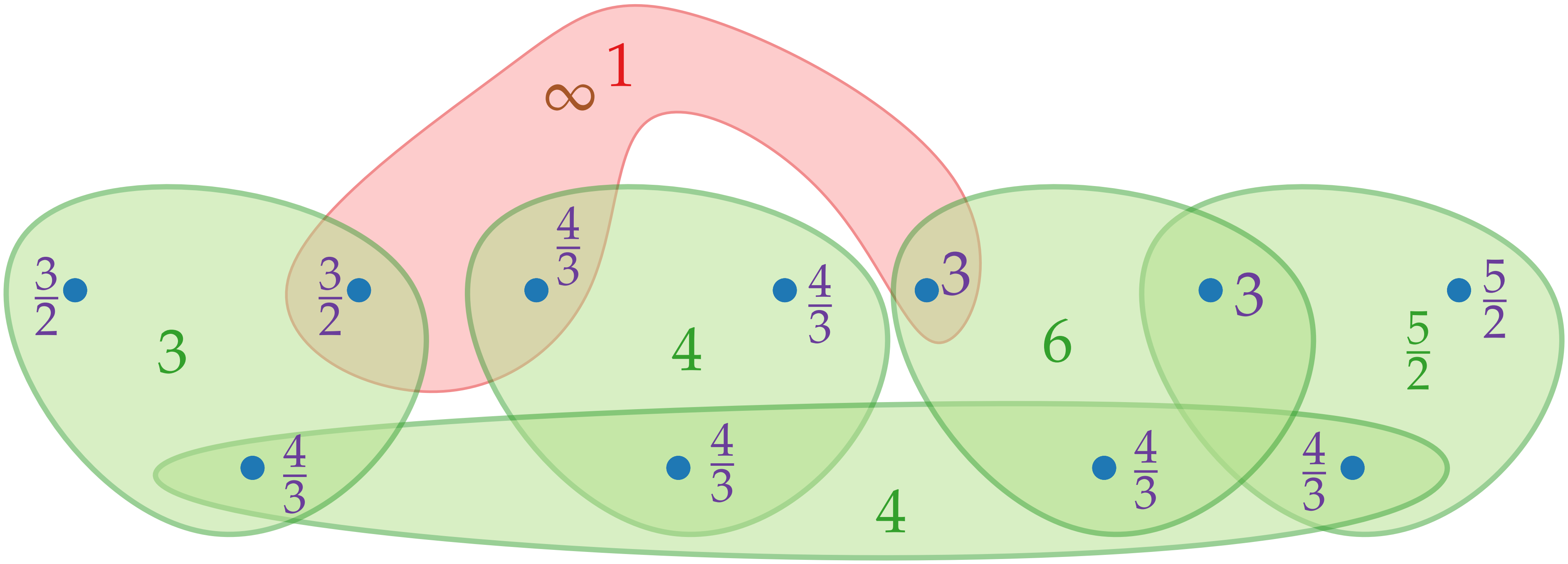
Set with k elements and cost c has unit cost $\frac{c}{k}$.

What happens if we "buy" a set?

Fix price of elements bought and recompute unit cost.

total cost: $\sum_{u \in U} \text{price}(u)$

Greedy: Always choose the set with the cheapest unit cost.



Greedy for SetCover

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

$C \leftarrow \emptyset$

$\mathcal{S}' \leftarrow \emptyset$

return $\mathcal{S}'$ `// Cover of` U

Greedy for SetCover

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

$C \leftarrow \emptyset$
$\mathcal{S}' \leftarrow \emptyset$
while $C \neq U$ **do**

return $\mathcal{S}'$ `// Cover of` U

Greedy for SetCover

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

```
C ← ∅
𝒮' ← ∅
while C ≠ U do
    S ← Set from 𝒮 that minimizes c(S)/|S \ C|

return 𝒮'                              // Cover of U
```

$C \leftarrow \emptyset$

$\mathcal{S}' \leftarrow \emptyset$

while $C \neq U$ **do**

  $S \leftarrow$ Set from $\mathcal{S}$ that minimizes $\frac{c(S)}{|S \setminus C|}$

return $\mathcal{S}'$ // Cover of U

Greedy for SetCover

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

$C \leftarrow \emptyset$
$\mathcal{S}' \leftarrow \emptyset$
while $C \neq U$ **do**
  $S \leftarrow$ Set from $\mathcal{S}$ that minimizes $\frac{c(S)}{|S \setminus C|}$
  foreach $u \in S \setminus C$ **do**
return $\mathcal{S}'$ `// Cover of` U

Greedy for SetCover

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

$C \leftarrow \emptyset$
$\mathcal{S}' \leftarrow \emptyset$
while $C \neq U$ **do**
  $S \leftarrow$ Set from $\mathcal{S}$ that minimizes $\frac{c(S)}{|S \setminus C|}$
  foreach $u \in S \setminus C$ **do**
    $\text{price}(u) \leftarrow \frac{c(S)}{|S \setminus C|}$
return $\mathcal{S}'$ `// Cover of` U

Greedy for SetCover

```
GreedySetCover(U, 𝒮, c)
  C ← ∅
  𝒮' ← ∅
  while C ≠ U do
      S ← Set from 𝒮 that minimizes c(S)/|S \ C|
      foreach u ∈ S \ C do
          price(u) ← c(S)/|S \ C|
      C ← C ∪ S

  return 𝒮'                                   // Cover of U
```

Greedy for SetCover

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

```
C ← ∅
S' ← ∅
while C ≠ U do
    S ← Set from 𝒮 that minimizes c(S)/|S \ C|
    foreach u ∈ S \ C do
        price(u) ← c(S)/|S \ C|
    C ← C ∪ S
    S' ← S' ∪ {S}
return S'                                  // Cover of U
```

$C \leftarrow \emptyset$

$\mathcal{S}' \leftarrow \emptyset$

while $C \neq U$ **do**

- $S \leftarrow$ Set from $\mathcal{S}$ that minimizes $\frac{c(S)}{|S \setminus C|}$
- **foreach** $u \in S \setminus C$ **do**
  - $\text{price}(u) \leftarrow \frac{c(S)}{|S \setminus C|}$
- $C \leftarrow C \cup S$
- $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{S\}$

return $\mathcal{S}'$ // Cover of U

Approximation Algorithms

Lecture 2:
SetCover and ShortestSuperString

Part III:
Analysis

Philipp Kindermann Summer Semester 2020

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\mathrm{price}(u_j) \leq$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\mathrm{price}(u_j) \leq$

Proof.

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then
$\text{price}(u_j) \leq$

Proof. Alg. buys $u_j \Rightarrow$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq$

Proof. Alg. buys $u_j \Rightarrow$

- $j - 1$ elements of S already bought

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq$

Proof. Alg. buys $u_j \Rightarrow$

- $j - 1$ elements of S already bought
- $\ell - j + 1$ elements of S not yet bought

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq$

Proof. Alg. buys $u_j \Rightarrow$

- $j - 1$ elements of S already bought
- $\ell - j + 1$ elements of S not yet bought
- unit cost for S:

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq$

Proof. Alg. buys $u_j \Rightarrow$

- $j - 1$ elements of S already bought
- $\ell - j + 1$ elements of S not yet bought
- unit cost for S: $c(S) / (\ell - j + 1)$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

Proof. Alg. buys $u_j \Rightarrow$

- $j - 1$ elements of S already bought
- $\ell - j + 1$ elements of S not yet bought
- unit cost for S: $c(S)/(\ell - j + 1)$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

Lemma. $\text{price}(S) := \sum_{i=1}^{\ell} \text{price}(u_i) \leq$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Proof.

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \le c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol.

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\text{OPT} = \sum_{i=1}^{m} c(S_i)$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\mathrm{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\mathrm{OPT} = \sum_{i=1}^{m} c(S_i)$
$\mathrm{price}(U) =$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\mathrm{price}(u_j) \le c(S)/(\ell - j + 1)$.

Lemma. $\mathrm{price}(S) := \sum_{i=1}^{\ell} \mathrm{price}(u_i) \le c(S) \cdot H_\ell$.

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\mathrm{OPT} = \sum_{i=1}^{m} c(S_i)$

$\mathrm{price}(U) = \sum_{u \in U} \mathrm{price}(u) \le$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \le c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\text{OPT} = \sum_{i=1}^{m} c(S_i)$
$\text{price}(U) = \sum_{u \in U} \text{price}(u) \le \sum_{i=1}^{m} \text{price}(S_i)$
$\le$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\text{OPT} = \sum_{i=1}^{m} c(S_i)$

$\text{price}(U) = \sum_{u \in U} \text{price}(u) \leq \sum_{i=1}^{m} \text{price}(S_i)$

$\leq \sum_{i=1}^{m} c(S_i) \cdot H_k =$

Analysis

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

Lemma. Let $S \in \mathcal{S}$ and $u_1, \ldots, u_\ell$ be the elements of S in the order they are covered ("bought") by GreedySetCover. Then $\text{price}(u_j) \leq c(S)/(\ell - j + 1)$.

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

Proof. Let $\{S_1, \ldots, S_m\}$ be opt. sol. $\text{OPT} = \sum_{i=1}^{m} c(S_i)$

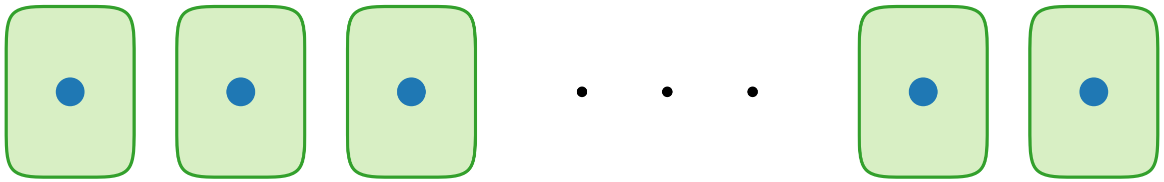
$$\text{price}(U) = \sum_{u \in U} \text{price}(u) \leq \sum_{i=1}^{m} \text{price}(S_i)$$
$$\leq \sum_{i=1}^{m} c(S_i) \cdot H_k = \text{OPT} \cdot H_k$$

Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.

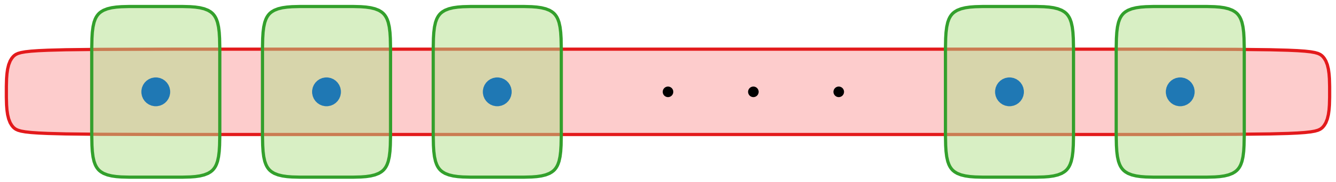
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in S and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



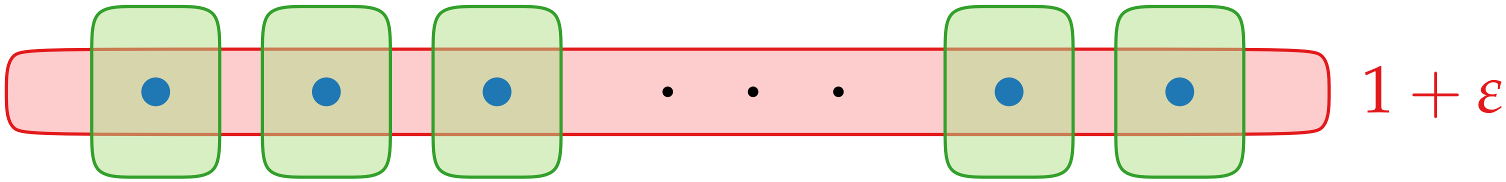
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



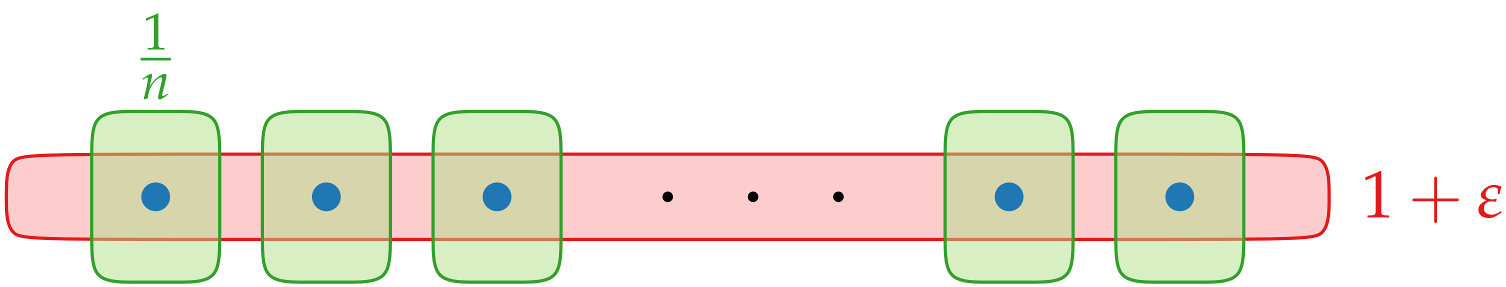
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



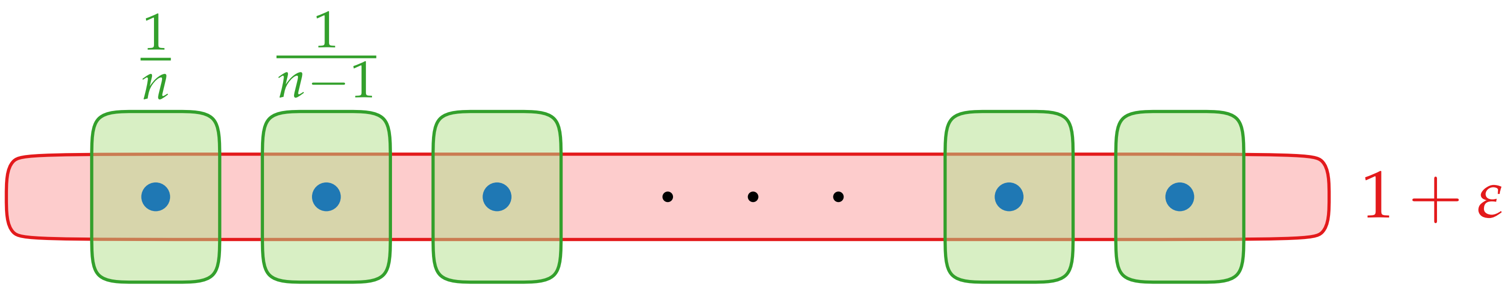
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



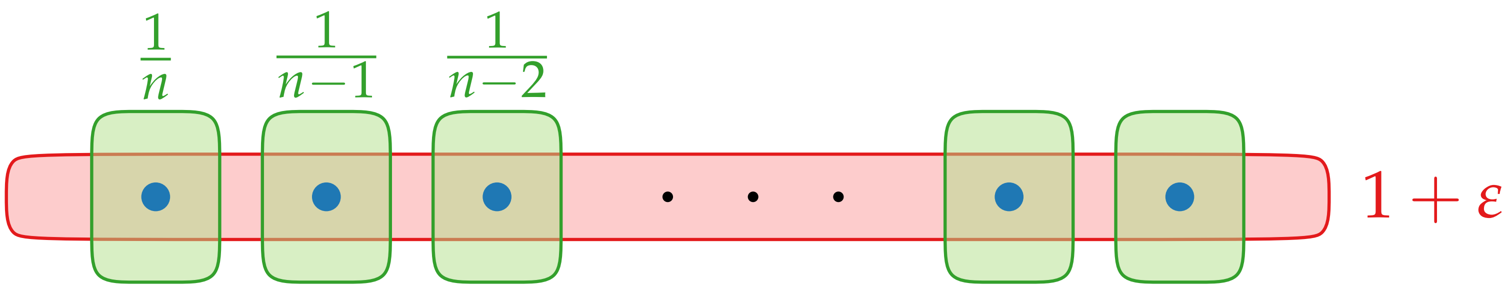
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



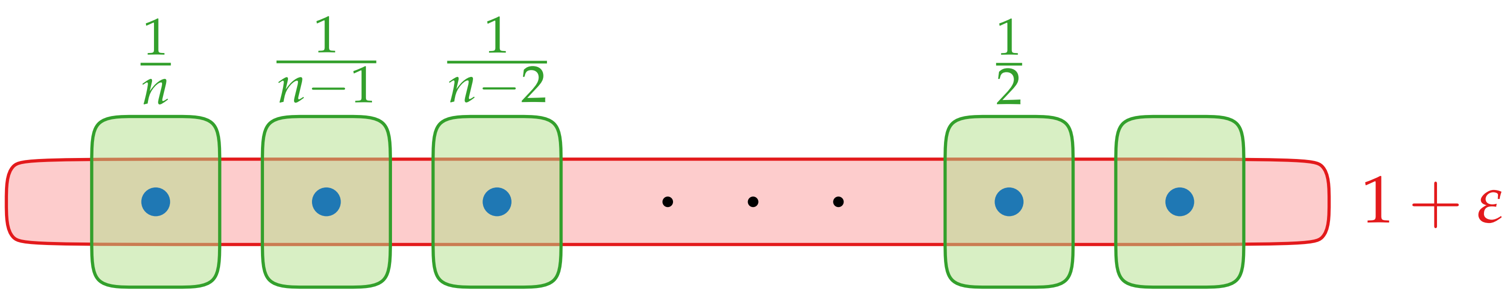
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



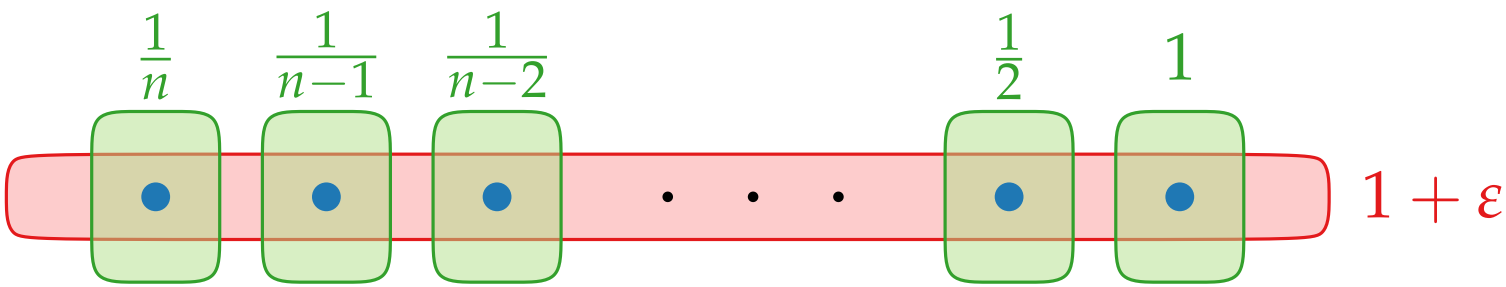
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



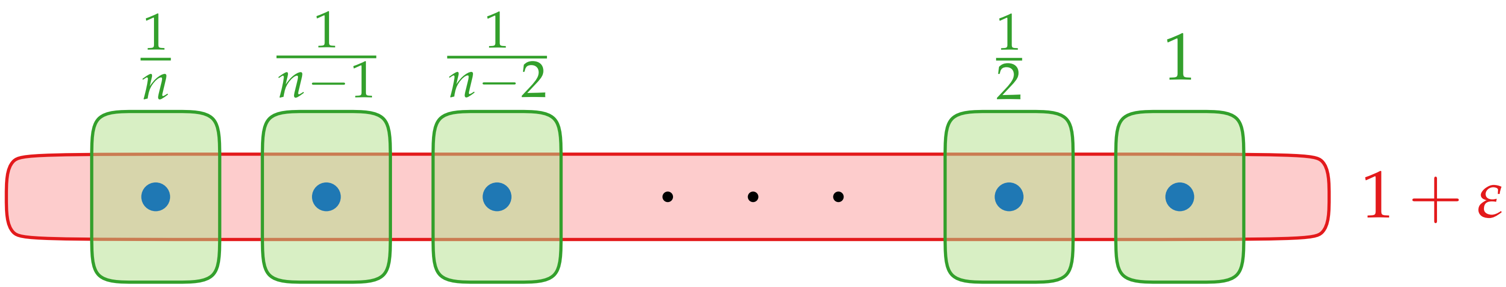
Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



Analysis sharp?

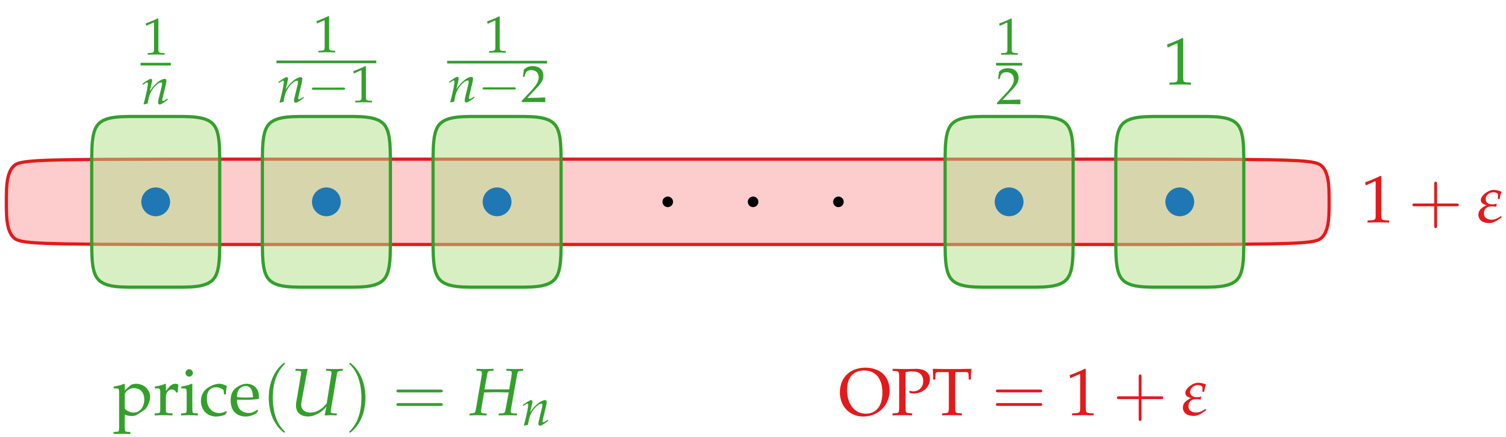
Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



price$(U) = H_n$ OPT $= 1 + \varepsilon$

Analysis sharp?

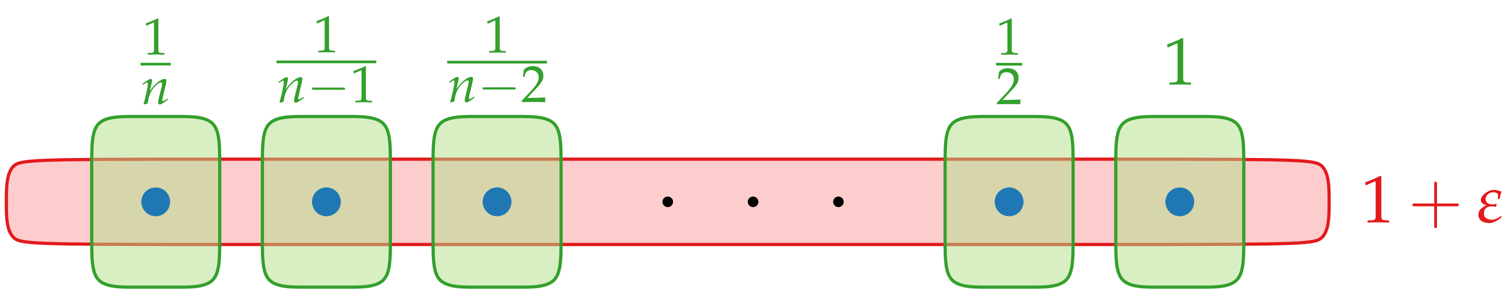
Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



better?

Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in $\mathcal{S}$ and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



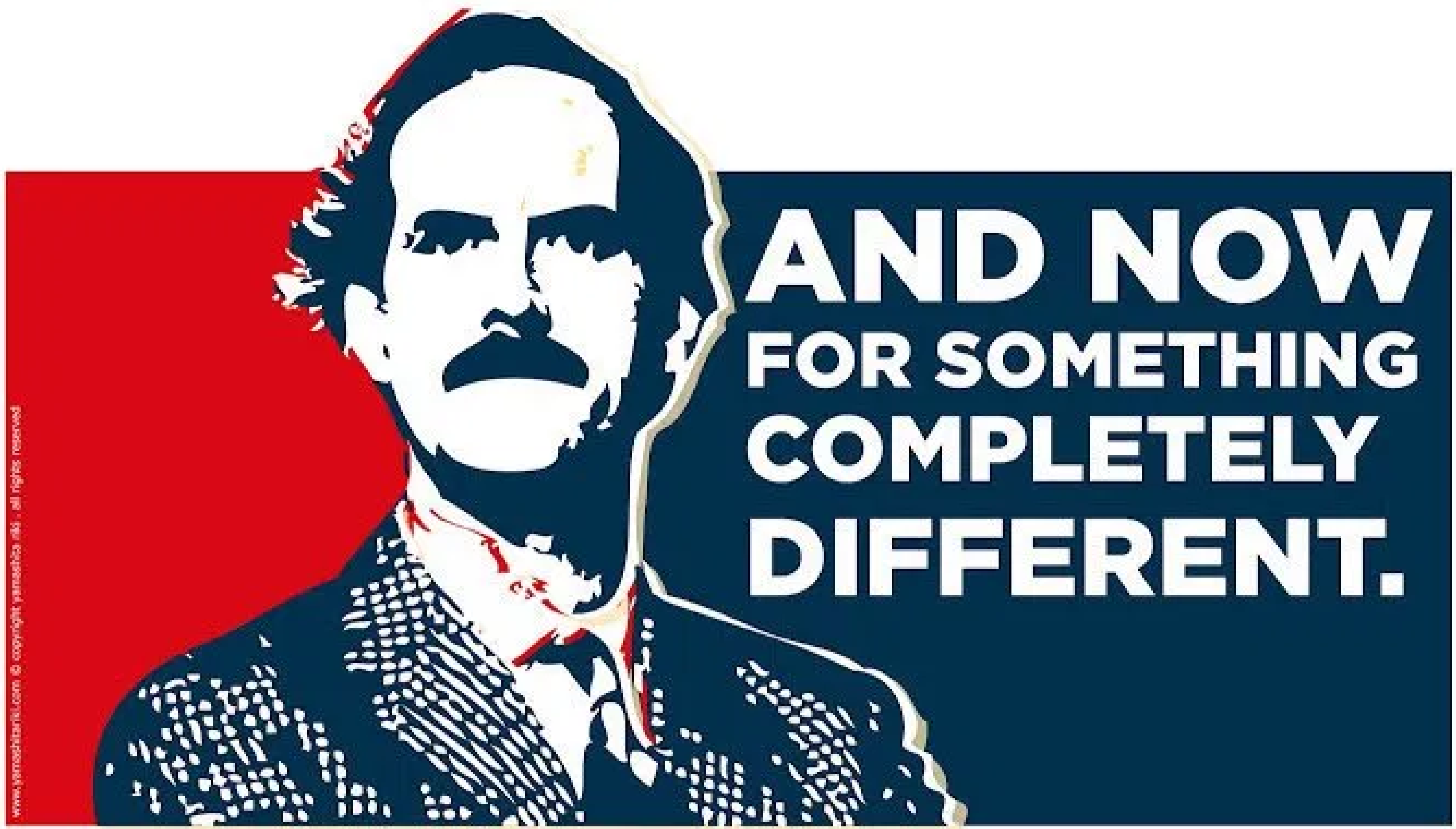
$\text{price}(U) = H_n$ $\quad$ $\text{OPT} = 1 + \varepsilon$

better?

SetCover cannot be approximated within factor $(1 - o(1)) \cdot \log(n)$ (unless P=NP)

Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in S and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



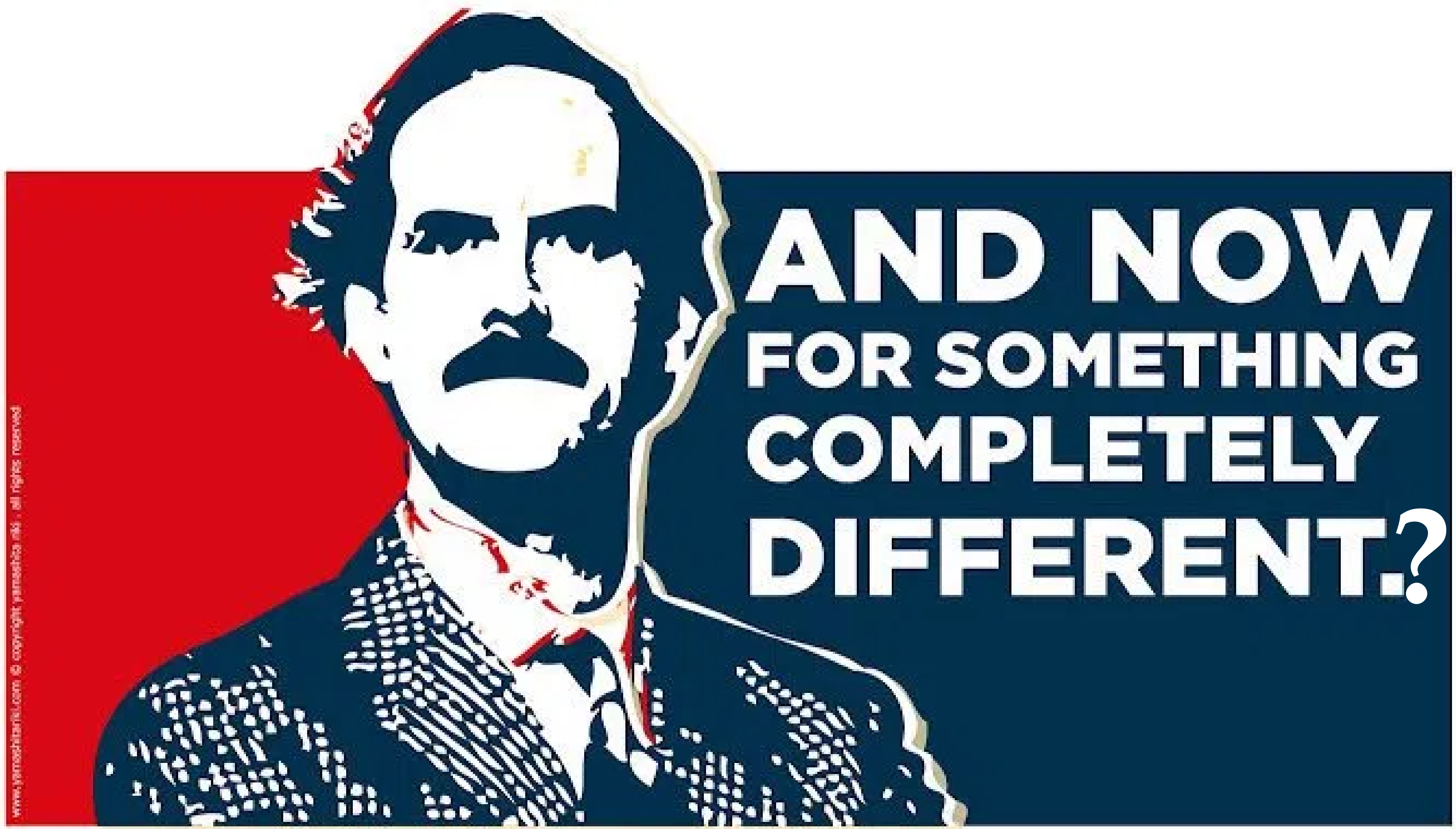
bette

SetC

$(1 -$

Analysis sharp?

Theorem. GreedySetCover is a factor-H_k-approximation algorithm for SetCover, where k is the cardinality of the largest set in S and $H_k := 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k} = O(\log k)$.



bette

SetC

$(1 -$

Approximation Algorithms

Lecture 2:
SetCover and ShortestSuperString

Part IV:
ShortestSuperString

Philipp Kindermann Summer Semester 2020

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

abc

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

abc
$\quad bcb$

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

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

ShortestSuperString (SSS)

Given a set $\{s_1, \dots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \dots, n$ is a *substring* of s.

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

⇓

$abcbaa$

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

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

"covers" all strings in U

$abcbaa$

abc

bcb

$cbaa$

ShortestSuperString (SSS)

Given a set $\{s_1, \ldots, s_n\} \subseteq \Sigma^+$ of strings over a finite alphabet Σ.
Find a **shortest string** s (*superstring*) such that each s_i, $i = 1, \ldots, n$ is a *substring* of s.

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

"covers" all strings in U

$abcbaa$

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

W.l.o.g.: No string s_i is a substring of any other string s_j.

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

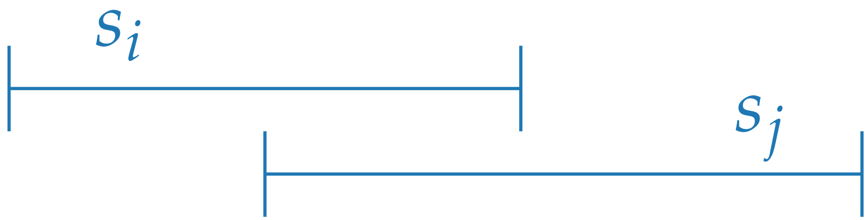
A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

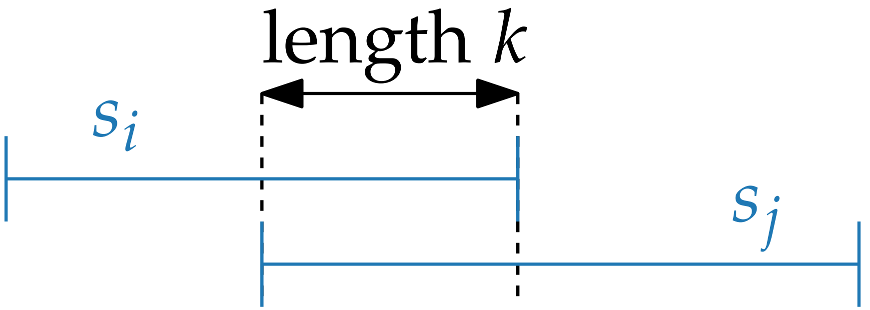


SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

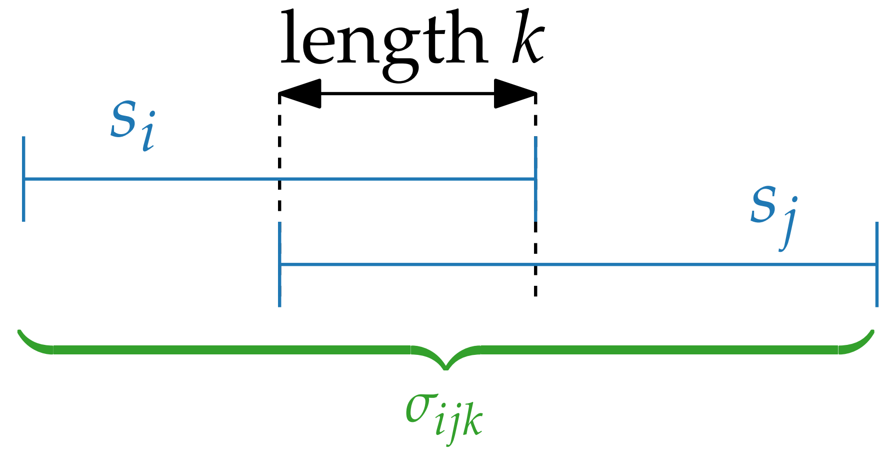


SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.



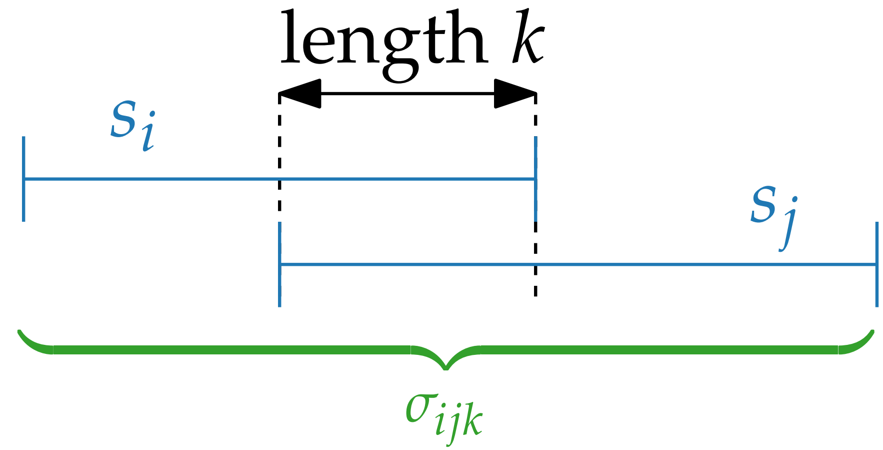
SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc



SSS as a SetCover Problem

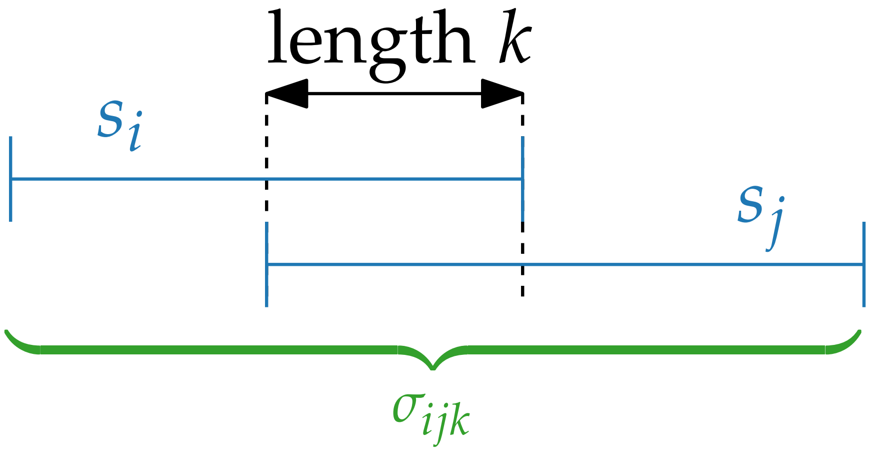
SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
ababc



SSS as a SetCover Problem

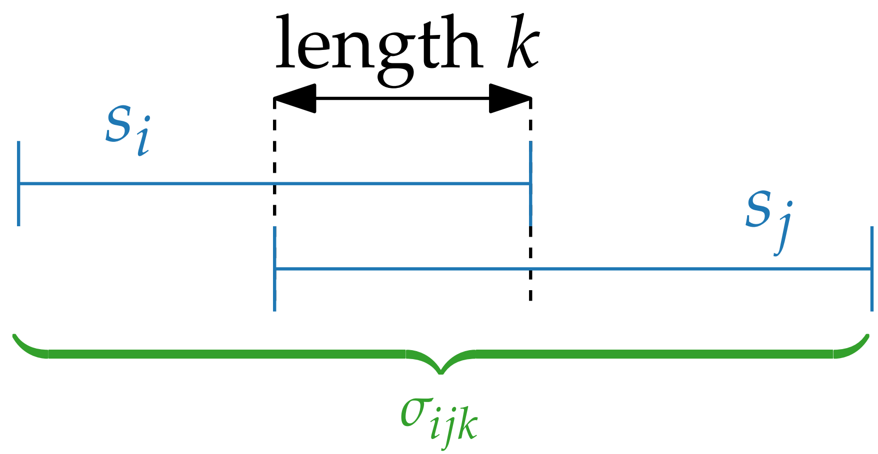
SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
  ababc



SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \dots, s_n\}$

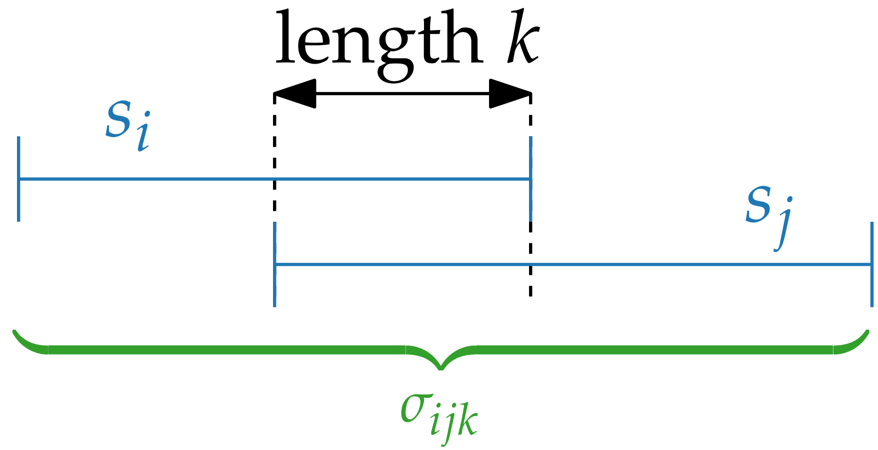
A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab

  ababc

σ_{ij2} : cabababc



SSS as a SETCOVER Problem

SETCOVER Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

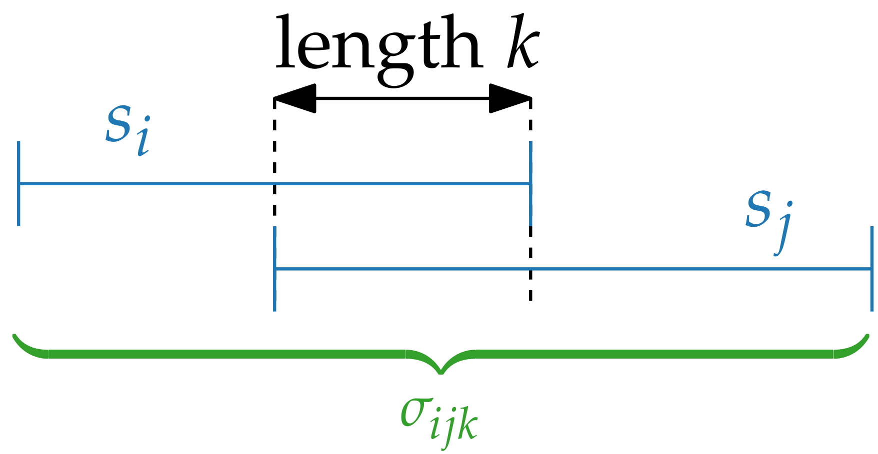
A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
  ababc

σ_{ij2} : cababab c

cabab
 ababc



SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

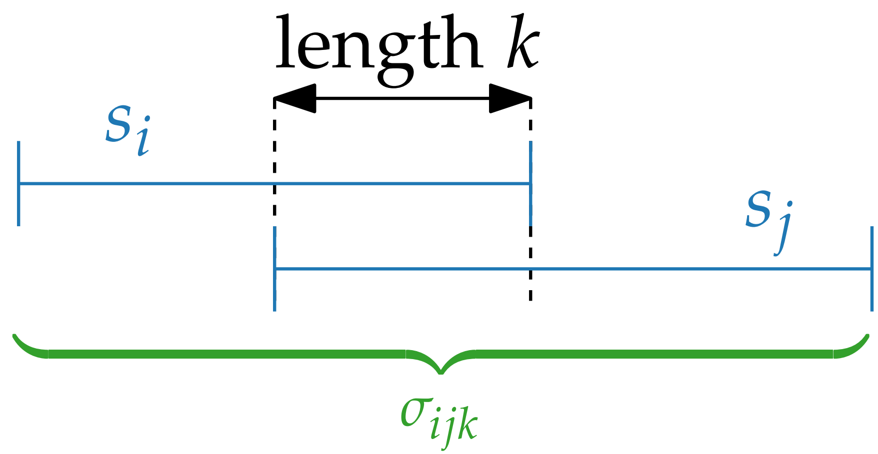
ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
ababc
σ_{ij2} : cababab c

cabab
ababc
σ_{ij4} : cababc



SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

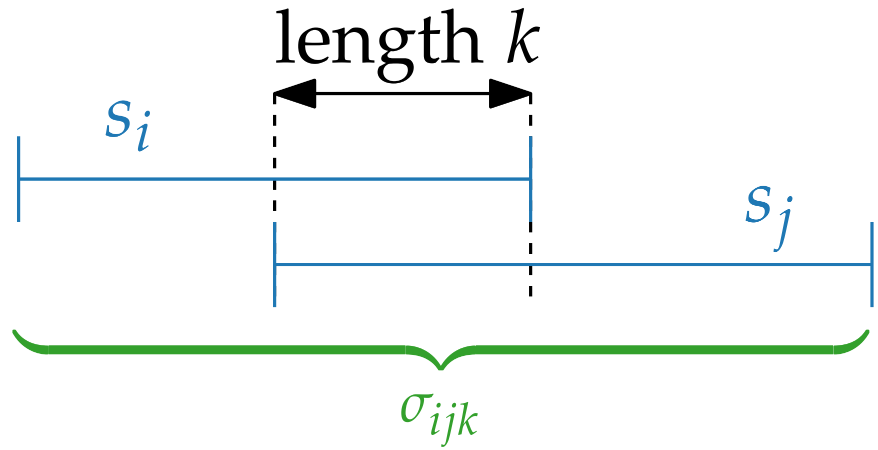
ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
ababc
σ_{ij2} : cababbabc

cabab
ababc
σ_{ij4} : cababc



$S(\sigma_{ijk}) = \{s \in U \mid s \text{ substring of } \sigma_{ijk}\}$ contains the elements of the ground set covered by σ_{ijk}.

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

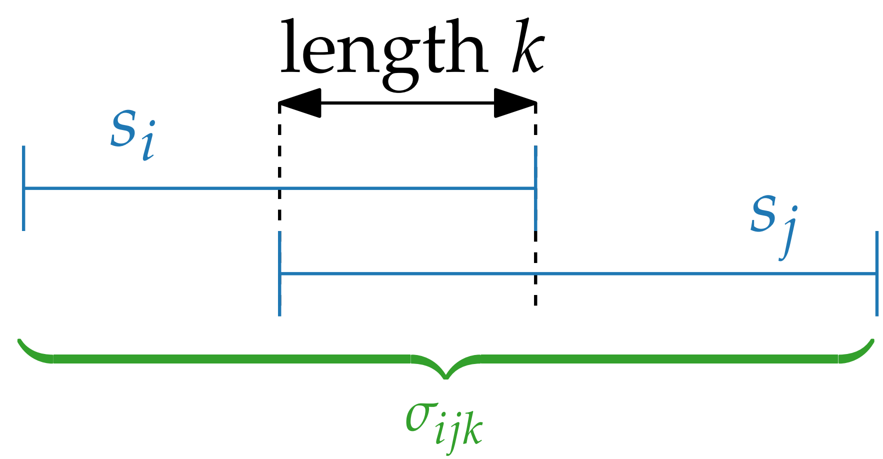
ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

s_i : cabab s_j : ababc

cabab
  ababc
σ_{ij2} : cababab c

 cabab
 ababc
σ_{ij4} : cababc



$S(\sigma_{ijk}) = \{s \in U \mid s \text{ substring of } \sigma_{ijk}\}$ contains the elements of the ground set covered by σ_{ijk}.

$c\left(S(\sigma_{ijk})\right) = |\sigma_{ijk}|$ (number of characters in σ_{ijk})

SSS as a SetCover Problem

SetCover Instance: ground set U, set family $\mathcal{S}$, costs c.

ground set $U := \{s_1, \ldots, s_n\}$

A string σ_{ijk} has prefix s_i and suffix s_j where s_i and s_j *overlap* on k characters.

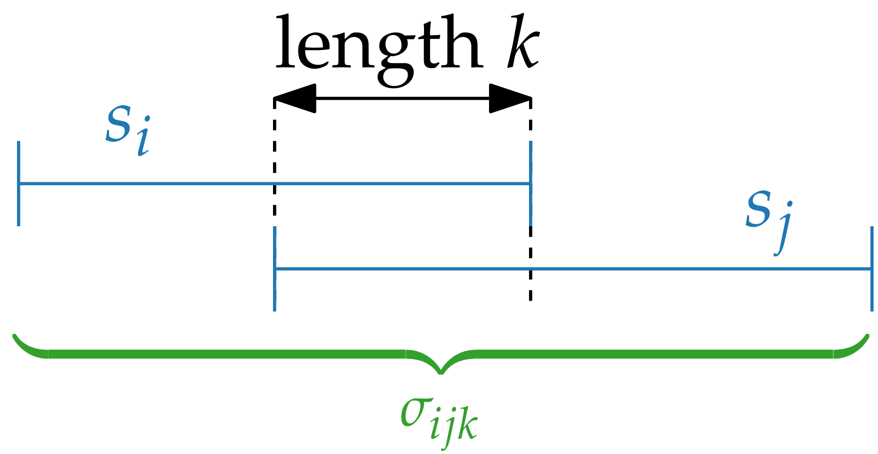
s_i : cabab s_j : ababc

cabab
  ababc

σ_{ij2} : cabababc

cabab
 ababc

σ_{ij4} : cababc



$S(\sigma_{ijk}) = \{s \in U \mid s \text{ substring of } \sigma_{ijk}\}$ contains the elements of the ground set covered by σ_{ijk}.

$c\left(S(\sigma_{ijk})\right) = |\sigma_{ijk}|$ (number of characters in σ_{ijk})

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

Approximation Algorithms

Lecture 2:
SetCover and ShortestSuperString

Part V:
Solving ShortestSuperString via SetCover

Philipp Kindermann Summer Semester 2020

Relating SSS and SetCover

Lemma. Let $\mathrm{OPT}_{\mathrm{SSS}}$ be the length of a shortest superstring of U and $\mathrm{OPT}_{\mathrm{SC}}$ be the minimum cost of the corresponding SetCover instance. Then:

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

Relating SSS and SetCover

Lemma. Let OPT_{SSS} be the length of a shortest superstring of U and OPT_{SC} be the minimum cost of the corresponding SetCover instance. Then:

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

Proof.

Consider an optimal set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ of U.

Relating SSS and SetCover

Lemma. Let $\mathrm{OPT}_{\mathrm{SSS}}$ be the length of a shortest superstring of U and $\mathrm{OPT}_{\mathrm{SC}}$ be the minimum cost of the corresponding SetCover instance. Then:

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

Proof.

Consider an optimal set cover $\{S(\pi_1), \dots, S(\pi_k)\}$ of U.

$s := \pi_1 \circ \cdots \circ \pi_k$ is a superstring of U of length

Relating SSS and SetCover

Lemma. Let $\mathrm{OPT}_{\mathrm{SSS}}$ be the length of a shortest superstring of U and $\mathrm{OPT}_{\mathrm{SC}}$ be the minimum cost of the corresponding SetCover instance. Then:

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

Proof.

Consider an optimal set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ of U.

$s := \pi_1 \circ \cdots \circ \pi_k$ is a superstring of U of length $\sum_{i=1}^{k} |\pi_i| = \sum_{i=1}^{k} c(S(\pi_i)) = \mathrm{OPT}_{\mathrm{SC}}$.

Relating SSS and SetCover

Lemma. Let $\mathrm{OPT}_{\mathrm{SSS}}$ be the length of a shortest superstring of U and $\mathrm{OPT}_{\mathrm{SC}}$ be the minimum cost of the corresponding SetCover instance. Then:

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

Proof.

Consider an optimal set cover $\{S(\pi_1), \dots, S(\pi_k)\}$ of U.

$s := \pi_1 \circ \dots \circ \pi_k$ is a superstring of U of length $\sum_{i=1}^{k} |\pi_i| = \sum_{i=1}^{k} c(S(\pi_i)) = \mathrm{OPT}_{\mathrm{SC}}$.

Thus, $\mathrm{OPT}_{\mathrm{SSS}} \leq |s| = \mathrm{OPT}_{\mathrm{SC}}$.

Relating SSS and SetCover

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

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.

s

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.
Construct set cover with cost$\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

s

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

leftmost occurence of a string $s_{b_1} \in U$.

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

Relating SSS and SetCover

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

Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

Relating SSS and SetCover

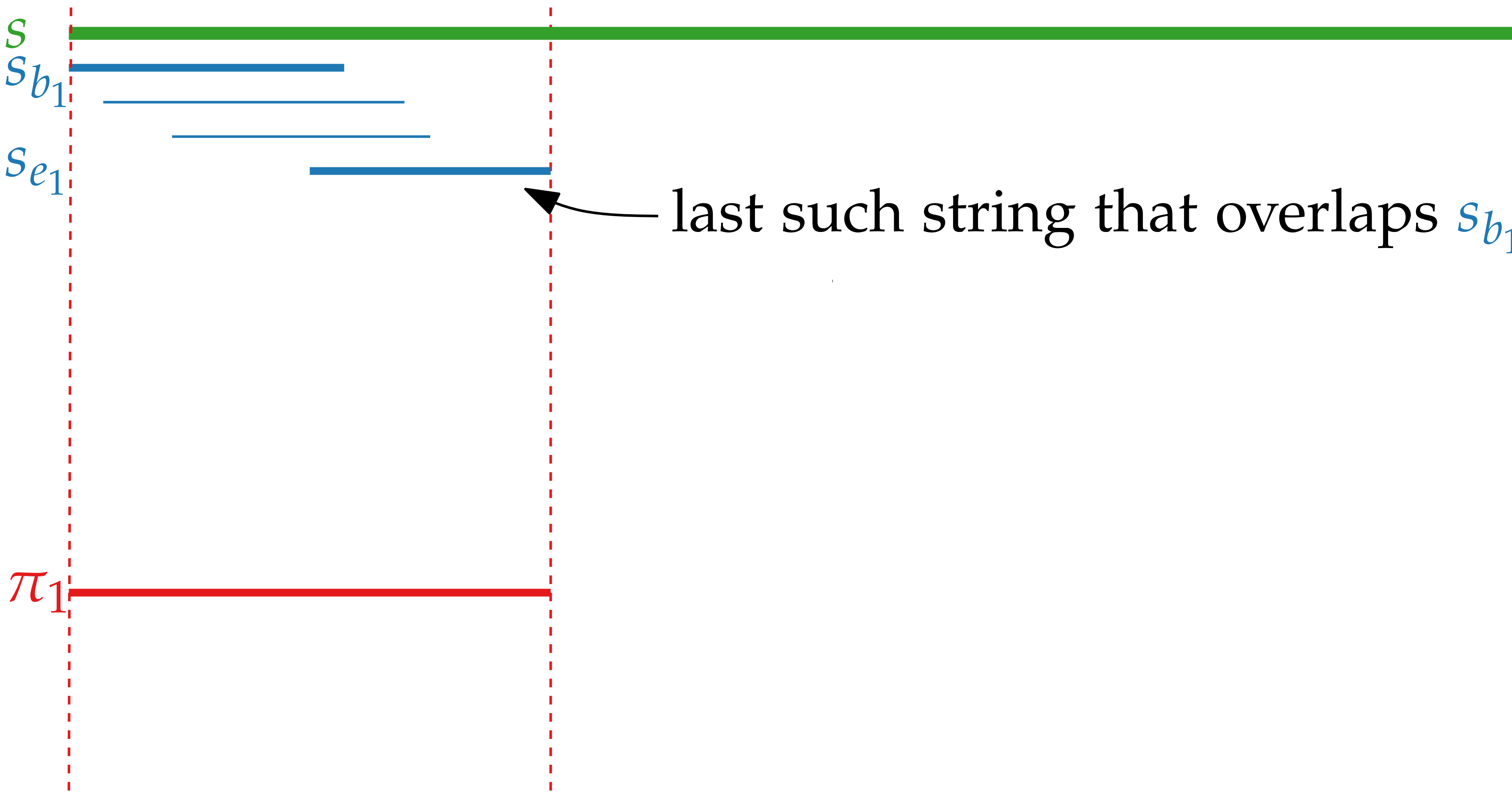
Lemma. $\mathrm{OPT}_{\mathrm{SC}} \leq 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$

Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.

Relating SSS and SetCover

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

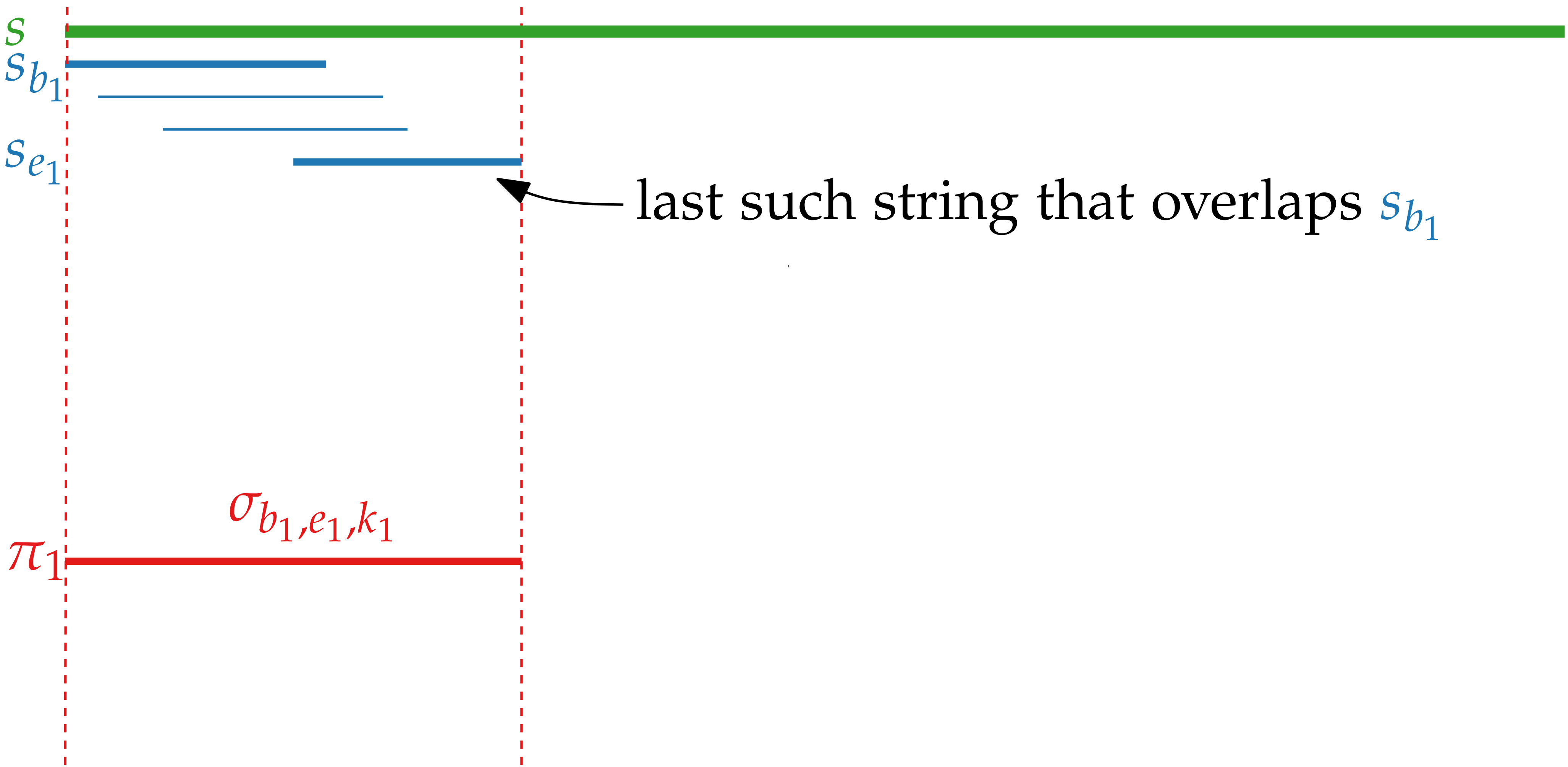
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \text{OPT}_{\text{SSS}}$.



Relating SSS and SetCover

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

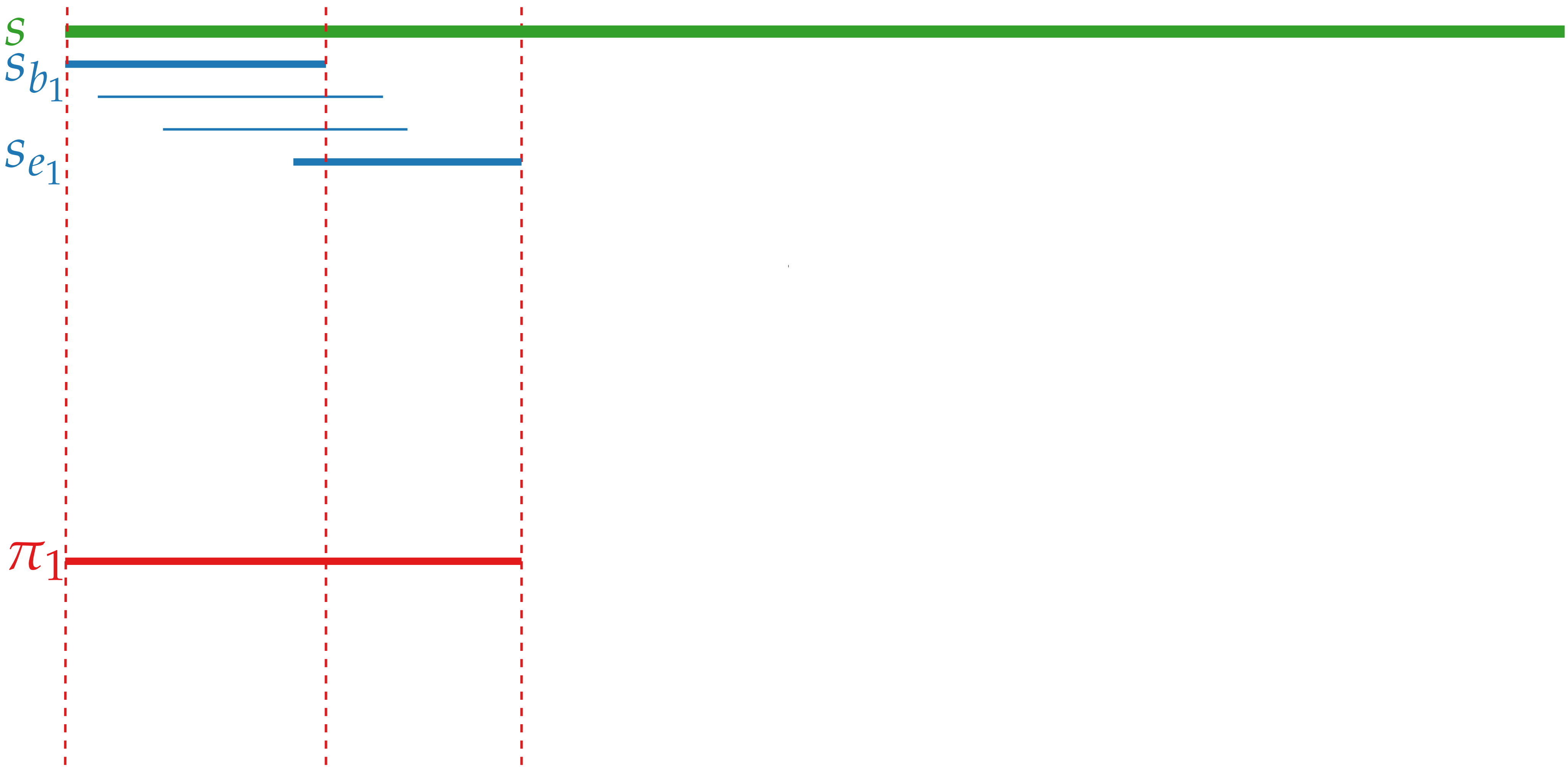
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

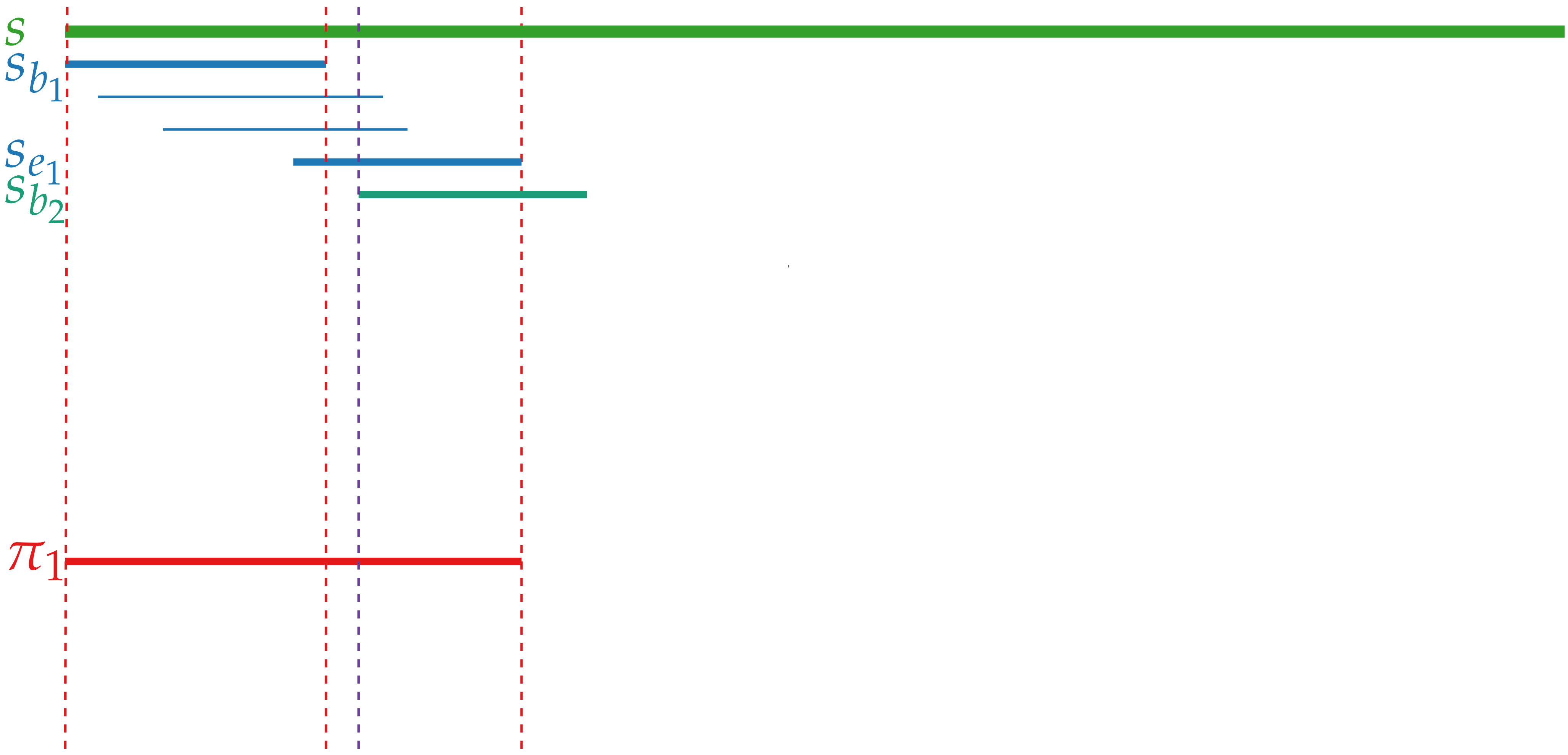
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

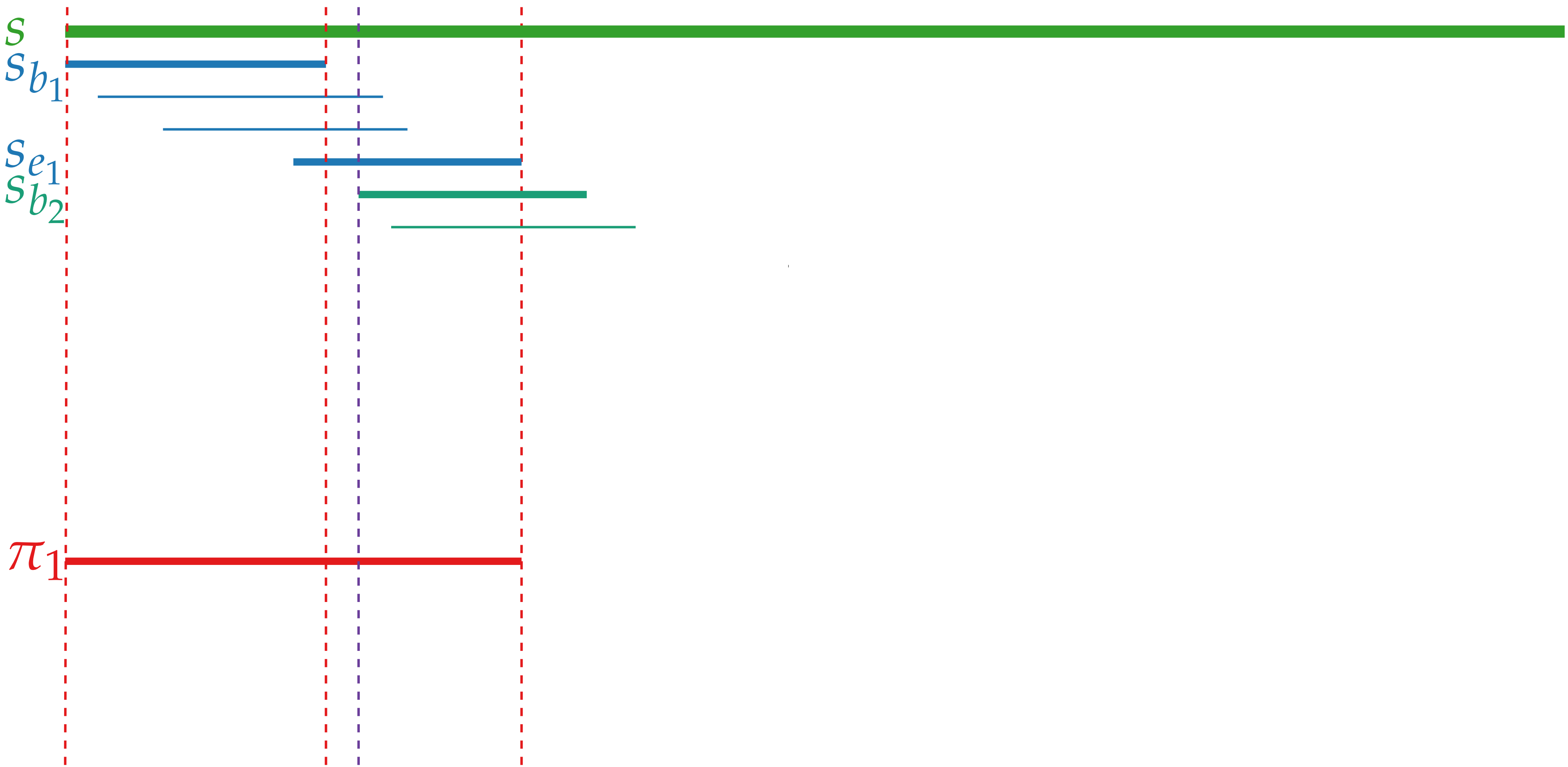
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

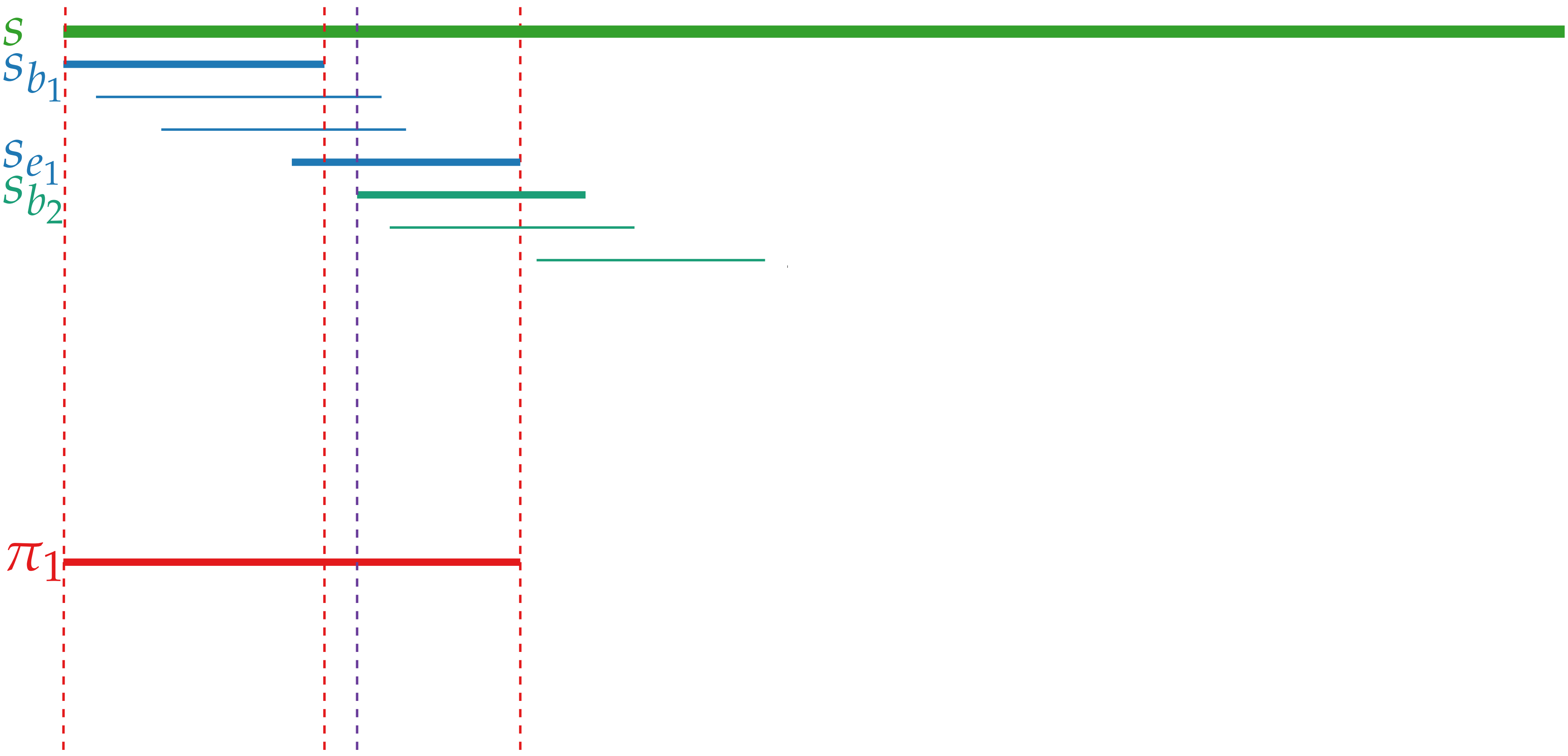
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

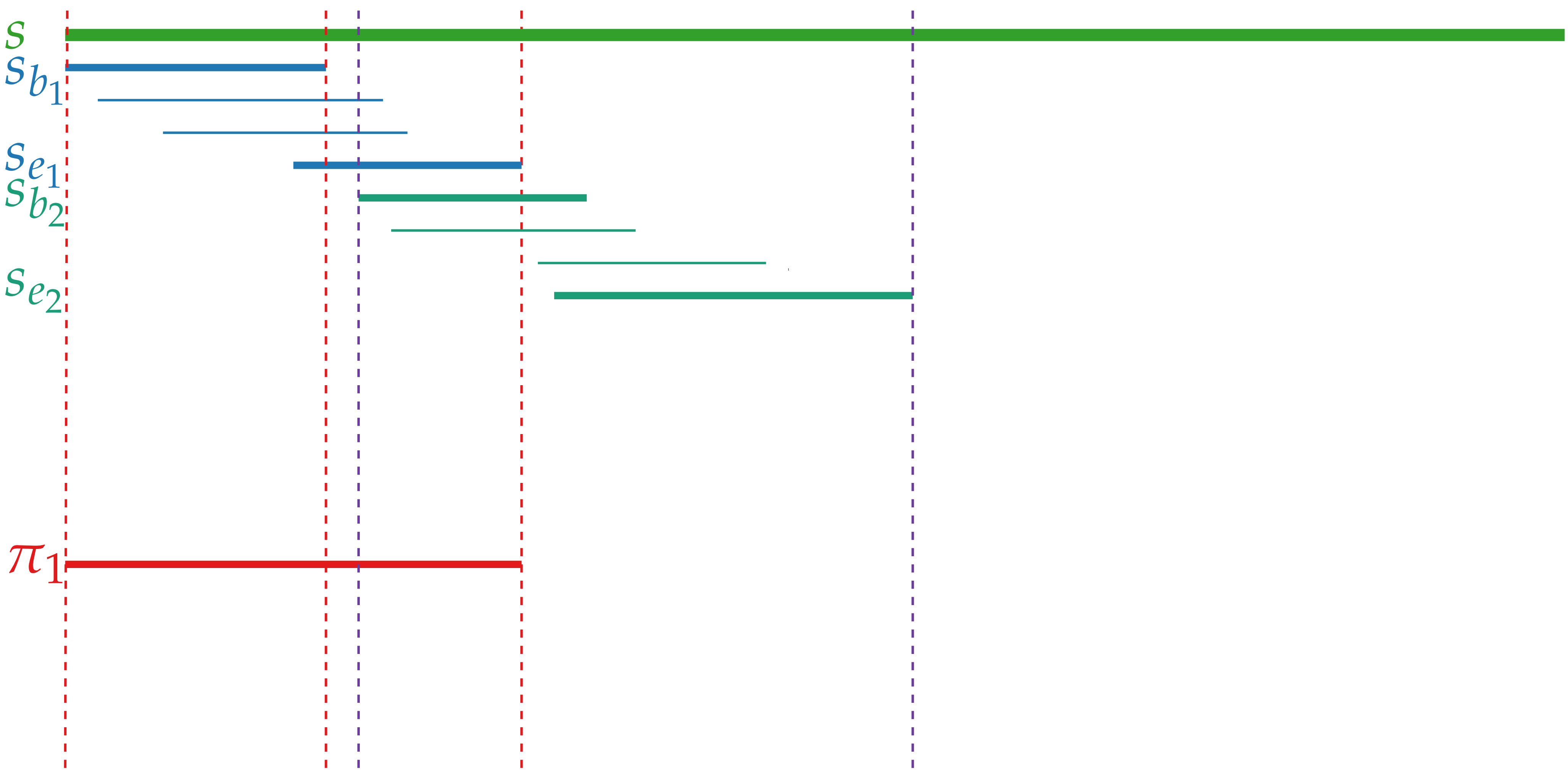
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

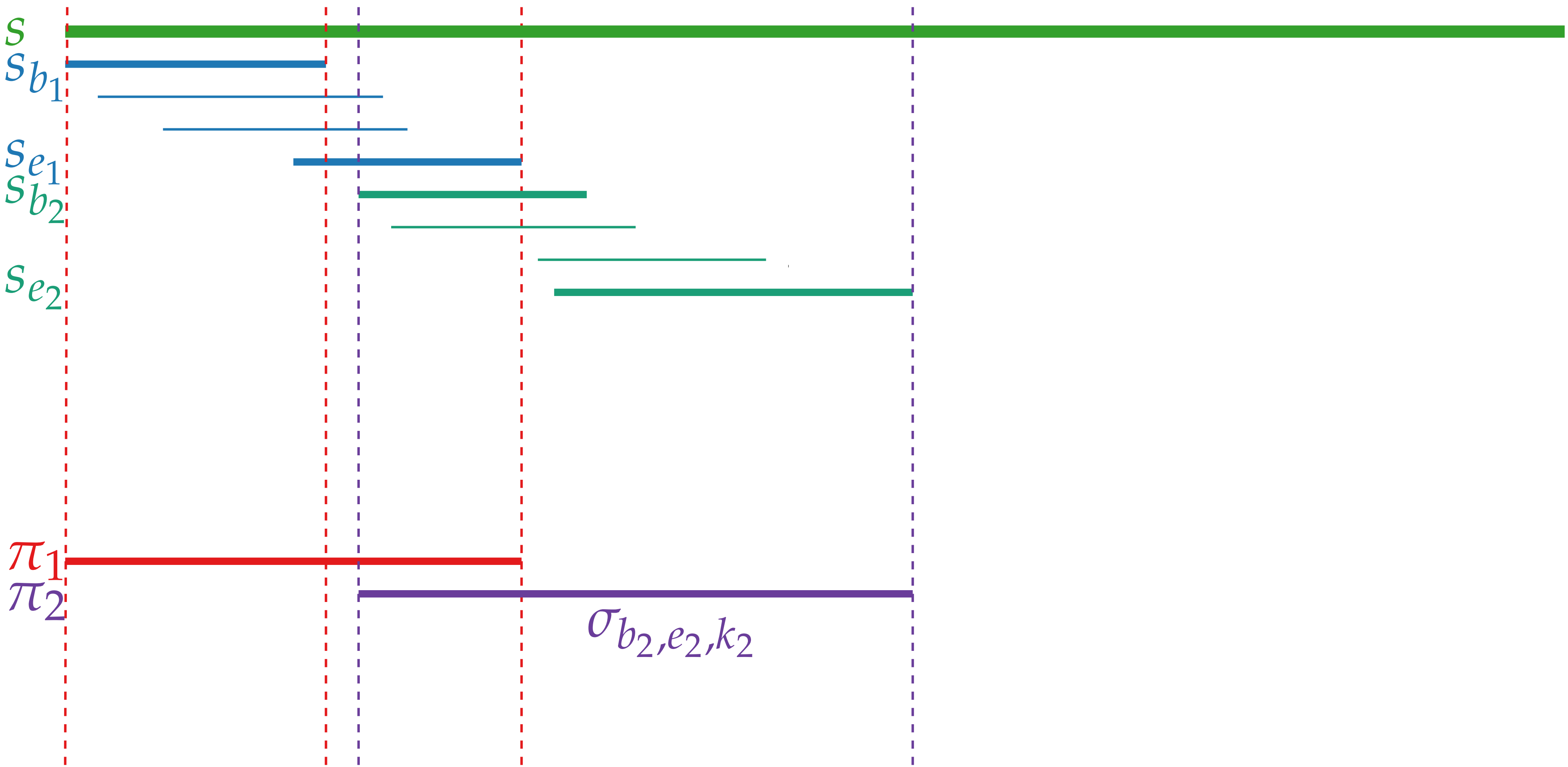
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \text{OPT}_{\text{SSS}}$.



Relating SSS and SetCover

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

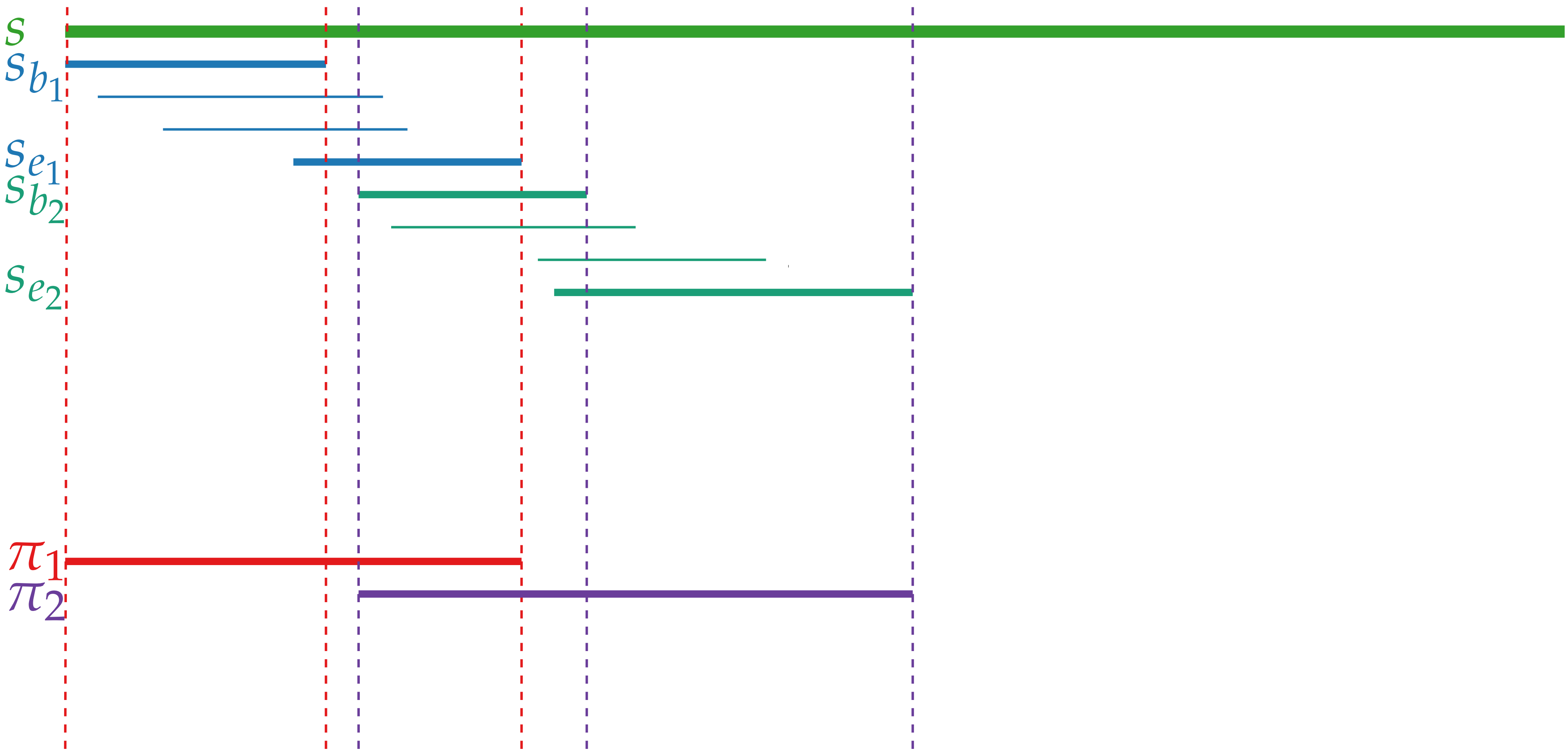
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

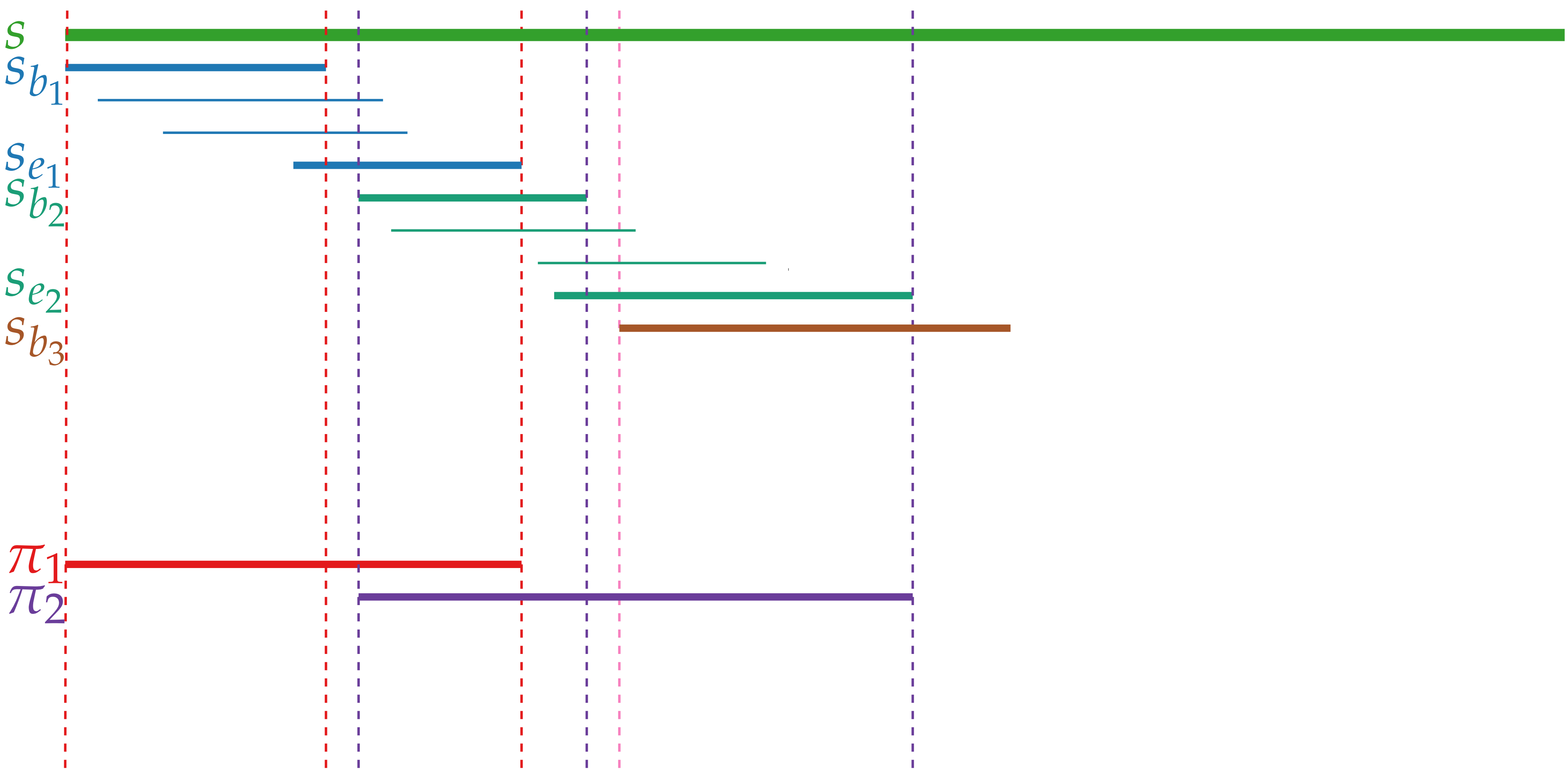
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

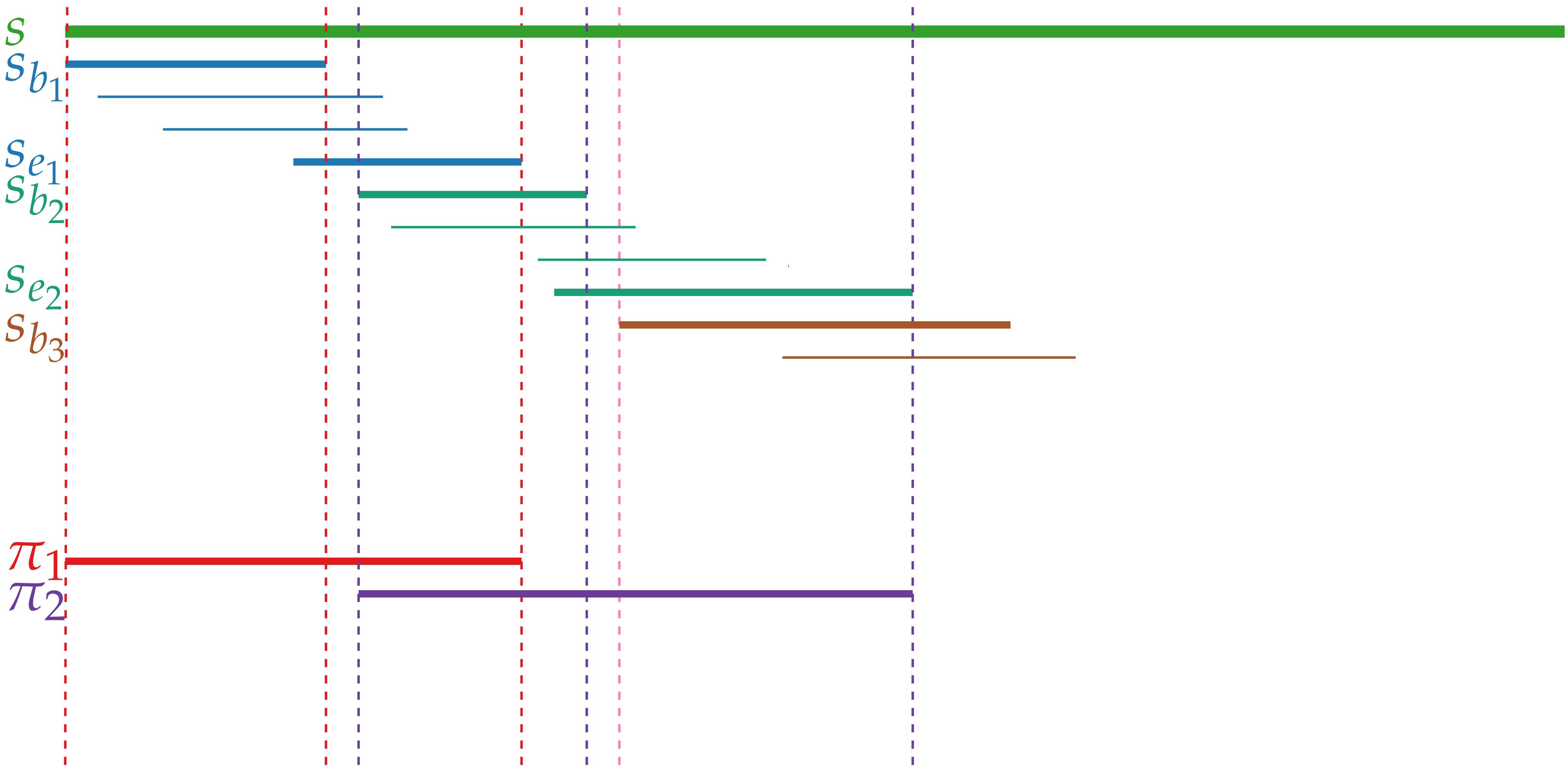
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

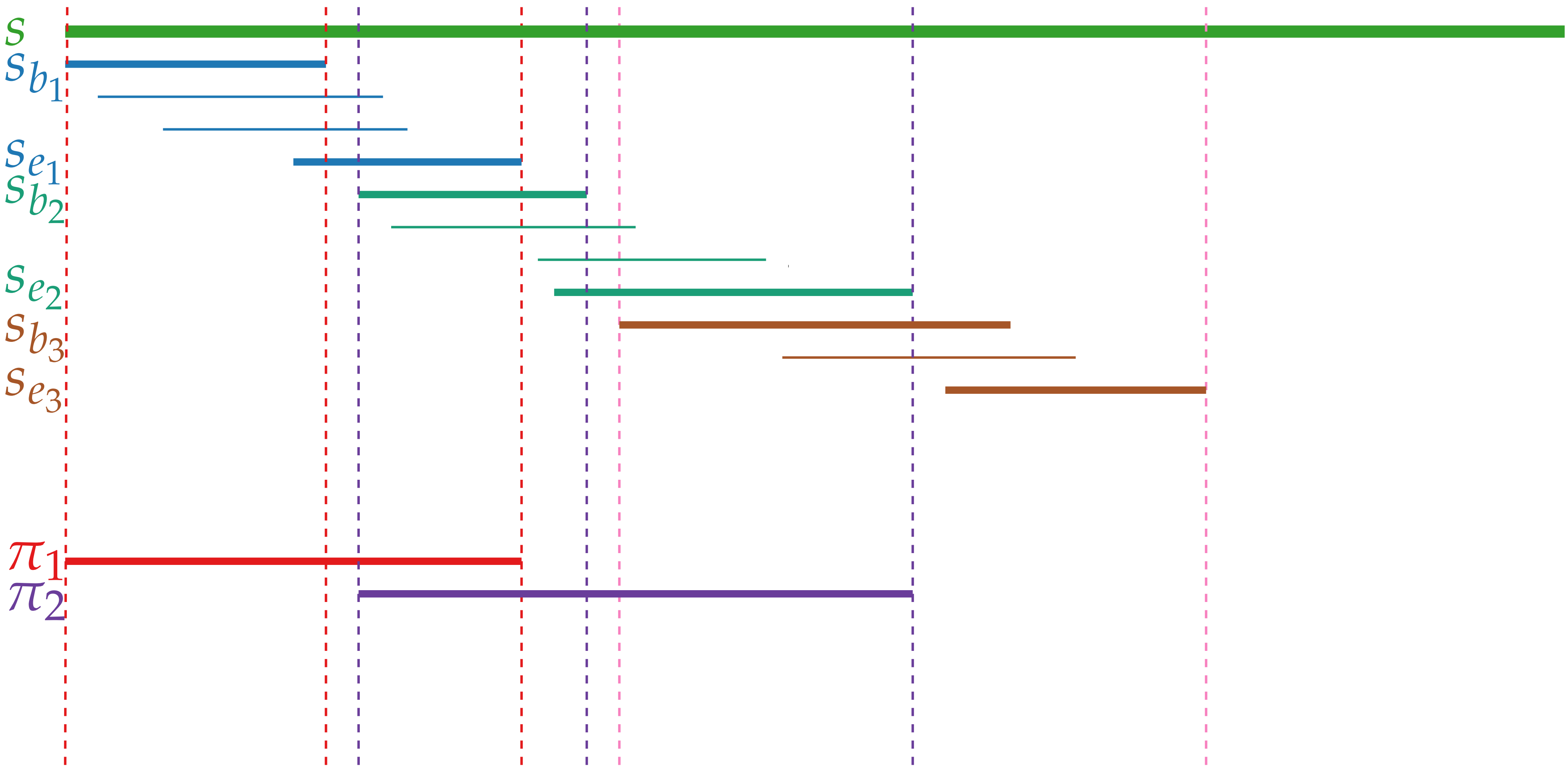
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

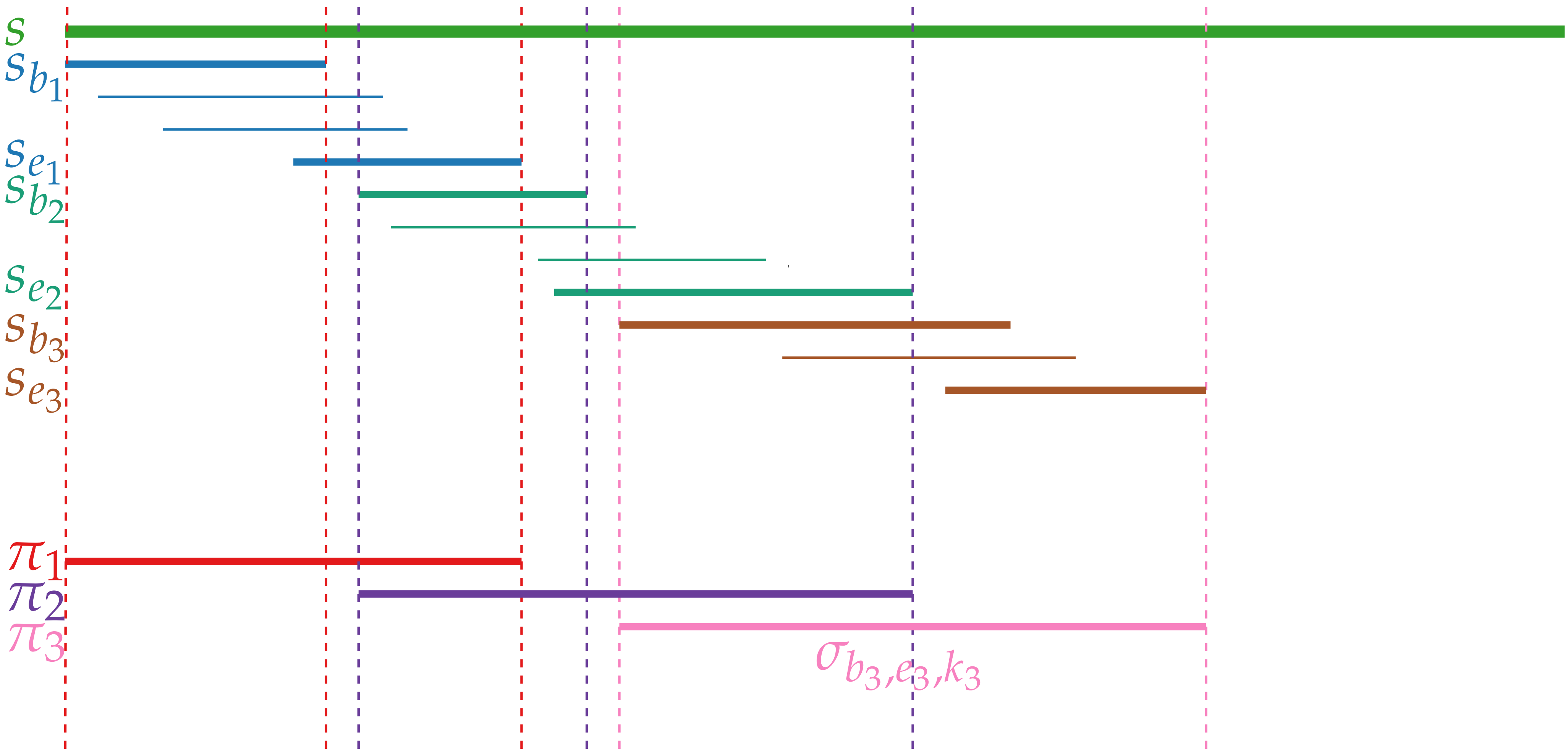
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

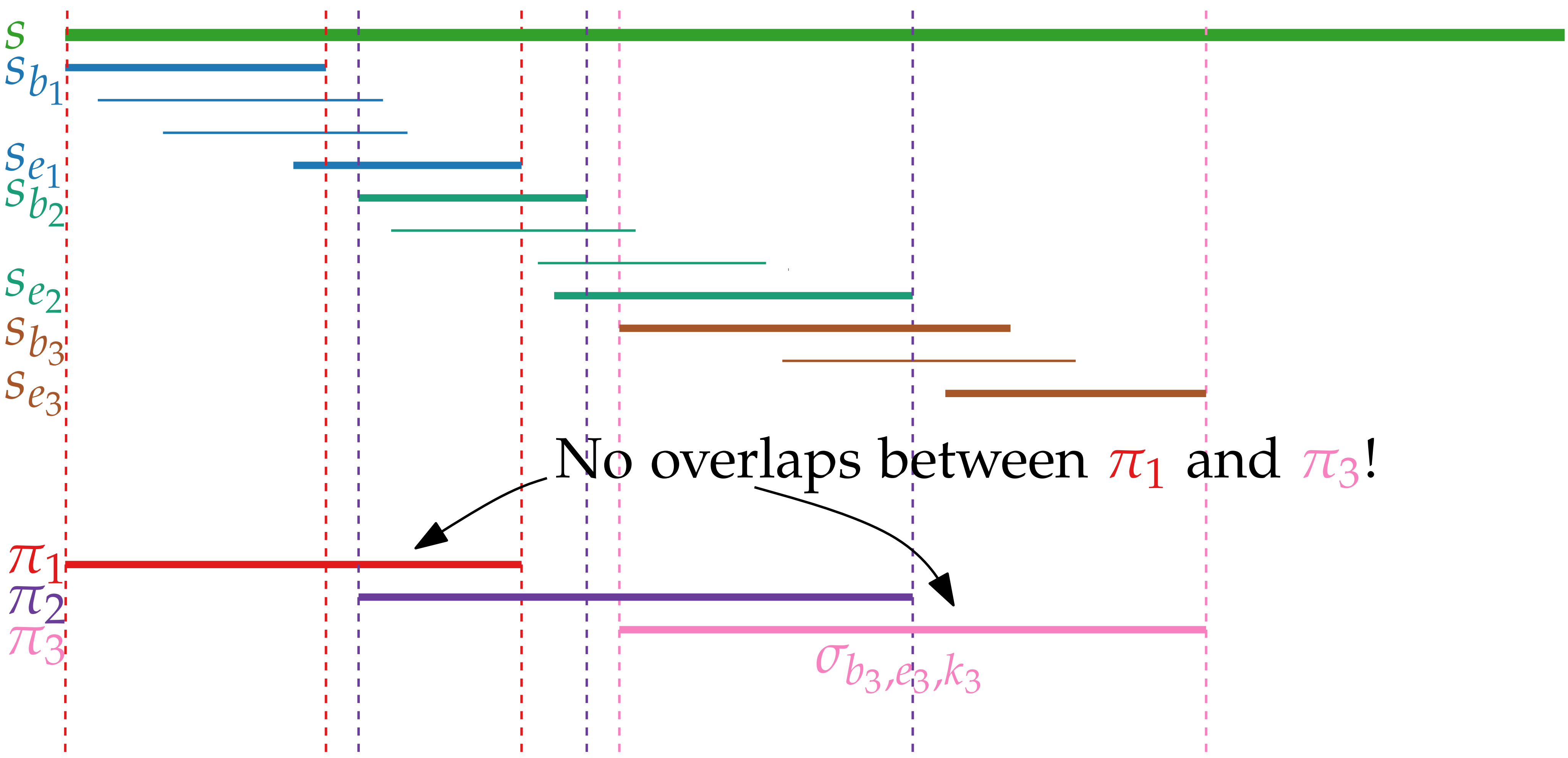
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

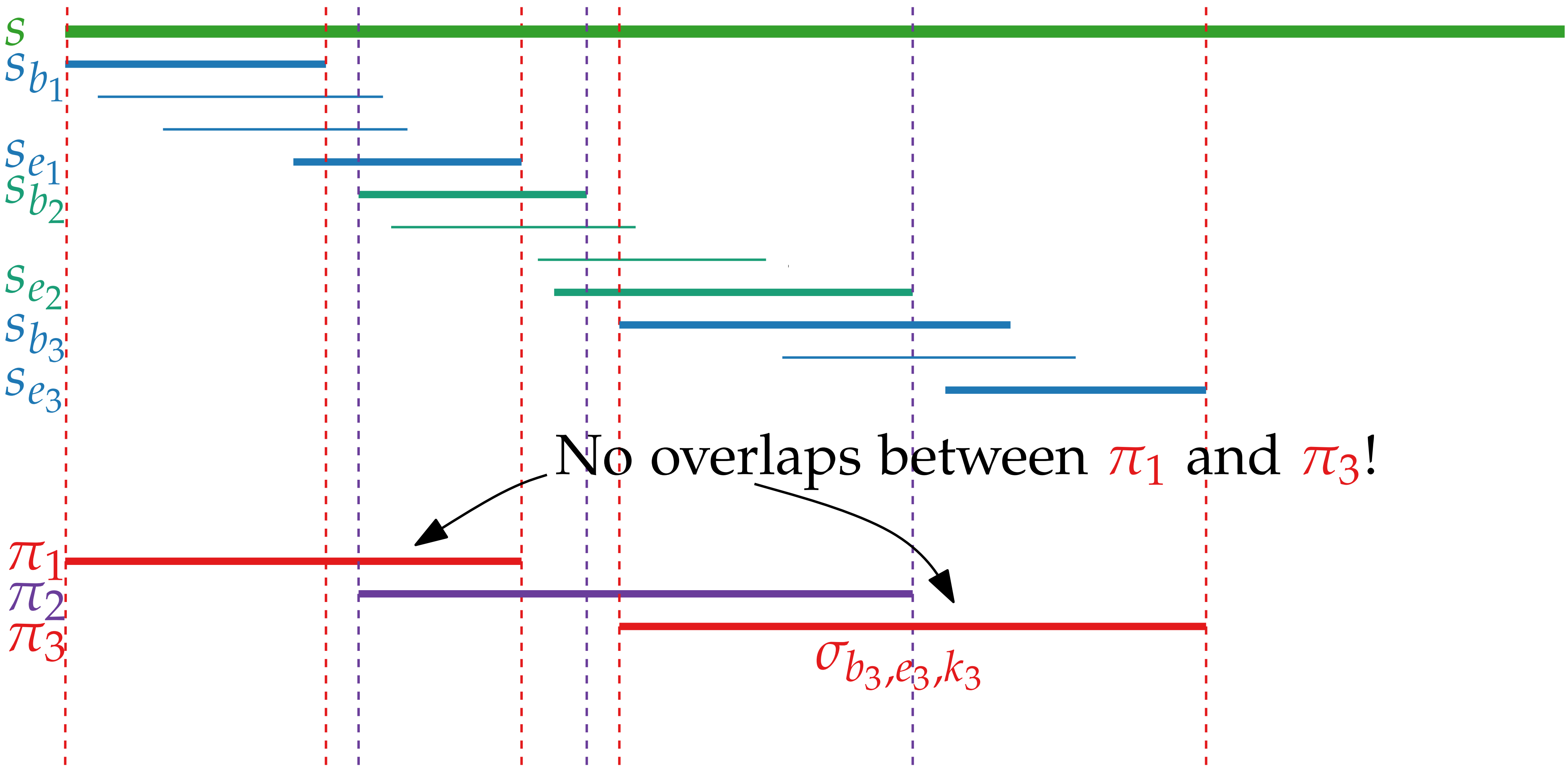
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

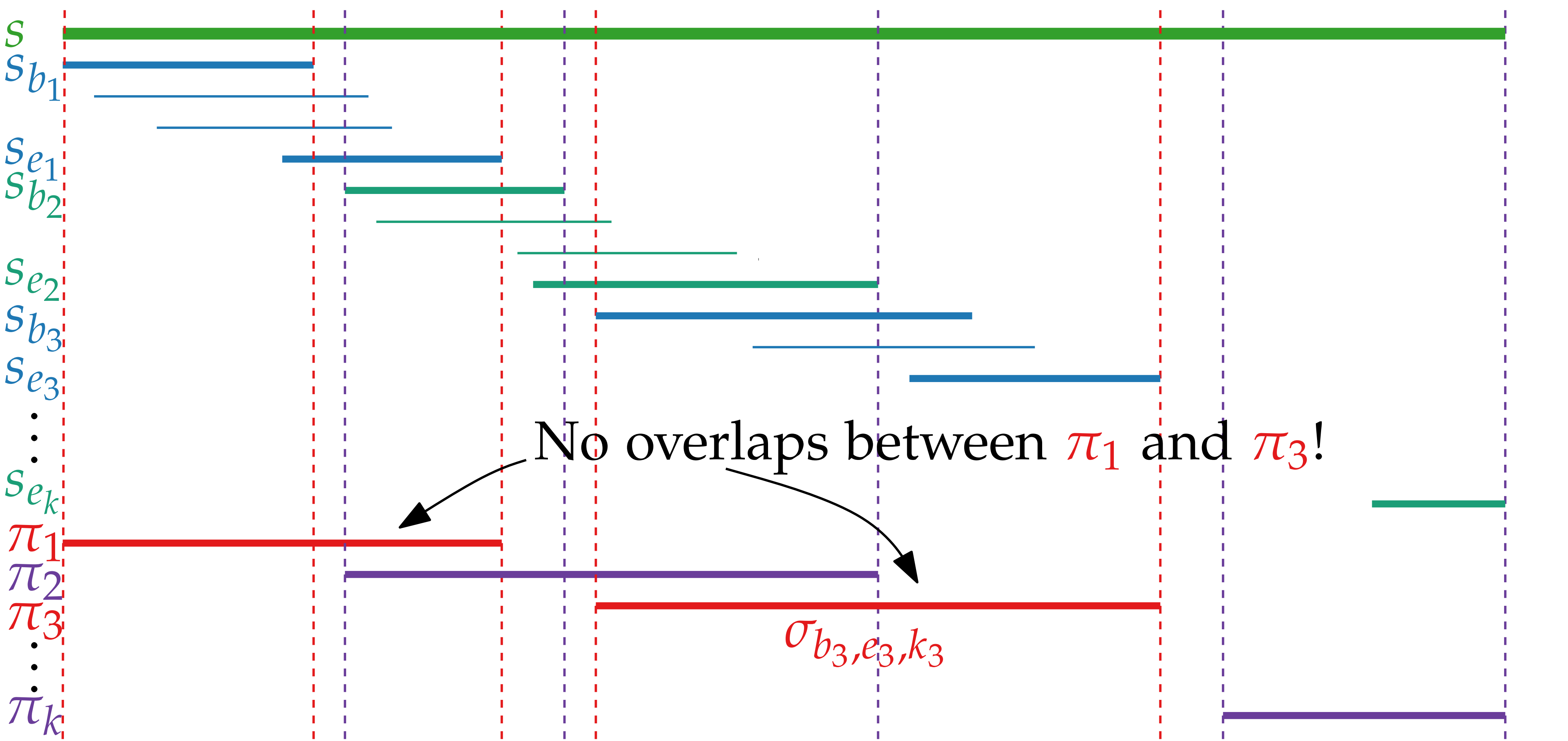
Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

Proof. Consider optimal superstring s.
Construct set cover with cost $\leq 2|s| = 2 \cdot \mathrm{OPT}_{\mathrm{SSS}}$.



Relating SSS and SetCover

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

Proof.

Each string $s_i \in U$ is a substring of some π_j.

Relating SSS and SetCover

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

Proof.

Each string $s_i \in U$ is a substring of some π_j.

$\{S(\pi_1), \ldots, S(\pi_k)\}$ is a solution for the SetCover instance with cost $\sum_i |\pi_i|$.

Relating SSS and SetCover

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

Proof.

Each string $s_i \in U$ is a substring of some π_j.

$\{S(\pi_1), \ldots, S(\pi_k)\}$ is a solution for the SetCover instance with cost $\sum_i |\pi_i|$.

Substrings π_j, π_{j+2} do not overlap.

Relating SSS and SetCover

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

Proof.

Each string $s_i \in U$ is a substring of some π_j.

$\{S(\pi_1), \ldots, S(\pi_k)\}$ is a solution for the SetCover instance with cost $\sum_i |\pi_i|$.

Substrings π_j, π_{j+2} do not overlap.

Each character lies in at most **two** (subsequent) substrings π_j und π_{j+1}.

Relating SSS and SetCover

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

Proof.

Each string $s_i \in U$ is a substring of some π_j.

$\{S(\pi_1), \ldots, S(\pi_k)\}$ is a solution for the SetCover instance with cost $\sum_i |\pi_i|$.

Substrings π_j, π_{j+2} do not overlap.

Each character lies in at most **two** (subsequent) substrings π_j und π_{j+1}.

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

Algorithm for SSS

1. Construct SetCover instance U, $\mathcal{S}$, c .

Algorithm for SSS

1. Construct SetCover instance U, $\mathcal{S}$, c .
2. Compute a set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ with algorithm GreedySetCover.

Algorithm for SSS

1. Construct SetCover instance U, $\mathcal{S}$, c .
2. Compute a set cover $\{S(\pi_1), \dots, S(\pi_k)\}$ with algorithm GreedySetCover.
3. Return $\pi_1 \circ \cdots \circ \pi_k$ as the superstring.

Algorithm for SSS

1. Construct SetCover instance U, $\mathcal{S}$, c .
2. Compute a set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ with algorithm GreedySetCover.
3. Return $\pi_1 \circ \cdots \circ \pi_k$ as the superstring.

Theorem. This algorithms is a factor-$2\mathcal{H}_n$-approximation algorithm for ShortestSuperString.

Algorithm for SSS

1. Construct SetCover instance $\mathcal{U}, \mathcal{S}, c$.
2. Compute a set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ with algorithm GreedySetCover.
3. Return $\pi_1 \circ \cdots \circ \pi_k$ as the superstring.

Theorem. This algorithms is a factor-$2\mathcal{H}_n$-approximation algorithm for ShortestSuperString.

better?

Algorithm for SSS

1. Construct SetCover instance U, $\mathcal{S}$, c .
2. Compute a set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ with algorithm GreedySetCover.
3. Return $\pi_1 \circ \cdots \circ \pi_k$ as the superstring.

Theorem. This algorithms is a factor-$2\mathcal{H}_n$-approximation algorithm for ShortestSuperString.

better?

The best-known approximation factor for ShortestSuperString is $\frac{71}{30} \approx 2.367$.

Algorithm for SSS

1. Construct SetCover instance $U, \mathcal{S}, c$.
2. Compute a set cover $\{S(\pi_1), \ldots, S(\pi_k)\}$ with algorithm GreedySetCover.
3. Return $\pi_1 \circ \cdots \circ \pi_k$ as the superstring.

Theorem. This algorithms is a factor-$2\mathcal{H}_n$-approximation algorithm for ShortestSuperString.

better?

The best-known approximation factor for ShortestSuperString is $\frac{71}{30} \approx 2.367$.

ShortestSuperString cannot be approximation within factor $\frac{333}{332} \approx 1.003$ (unless P=NP).