

Advanced topics on Algorithms

Luciano Gualà

www.mat.uniroma2.it/~guala/

What: 3 topics, 4 lectures per topic

approximation algorithms:

- well-established field
- widely used approach for (NP-)hard problems
- **cool techniques:** rounding, dual-fitting, primal-dual approach

parameterized algorithms:

- multivariate analysis of algorithms
- refined notions of efficiency and hardness
- **cool techniques:** color coding, kernelization, treewidth

algorithms for massive data:

- primitives for most data mining applications
- sublinear time and space algorithms
- **cool techniques:** sampling, sketches, locality sensitive hashing

How (to get credits)

- attend lectures
- final oral exam and/or class presentation (of uncovered material)

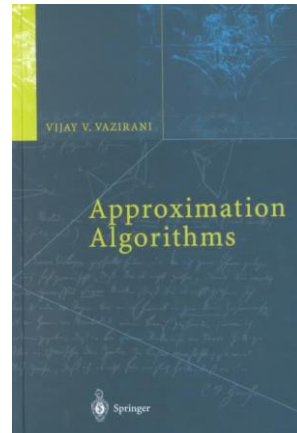
Why

- **expand your background:** wider view of the huge world of algorithms
- **useful:** be better theorists and practitioners
- **fun:** amazing material and techniques

any question?

Approximation algorithms: Episode I (pilot)

main reference:



Def.

An α -approximation algorithm for an optimization problem is a polynomial-time algorithm that for all instances of the problem produces a solution whose value is within a factor of α the value of an optimal solution.

α : approximation ratio or approximation factor

minimization problem:

- $\alpha \geq 1$
- for each returned solution x , $\text{cost}(x) \leq \alpha \text{ OPT}$

maximization problem:

- $\alpha \leq 1$
- for each returned solution x , $\text{value}(x) \geq \alpha \text{ OPT}$

minimum Vertex Cover
problem

min cardinality Vertex Cover problem

Input:

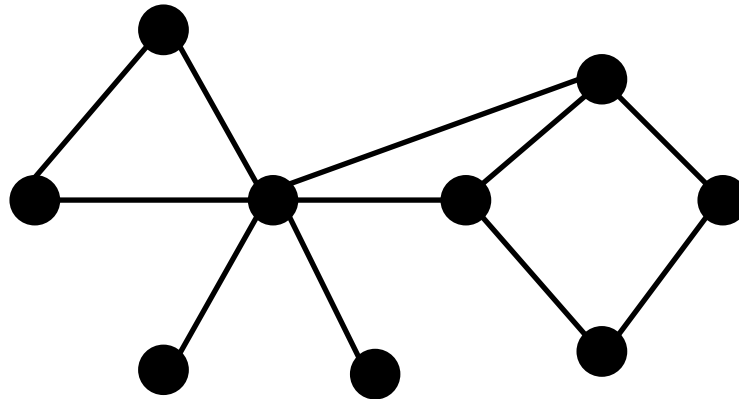
an undirected graph $G=(V,E)$

Feasible solution:

$U \subseteq V$ such that every edge $(u,v) \in E$ is covered, i.e. $u \in U$ or $v \in U$

measure (min):

cardinality of U



min cardinality Vertex Cover problem

Input:

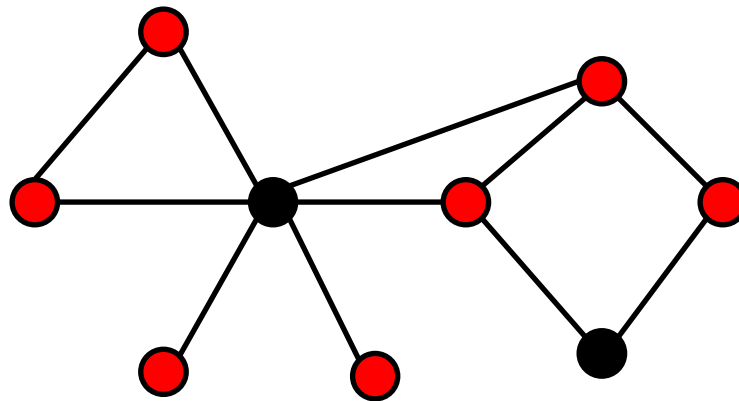
an undirected graph $G=(V,E)$

Feasible solution:

$U \subseteq V$ such that every edge $(u,v) \in E$ is covered, i.e. $u \in U$ or $v \in U$

measure (min):

cardinality of U



a vertex cover of size 7

min cardinality Vertex Cover problem

Input:

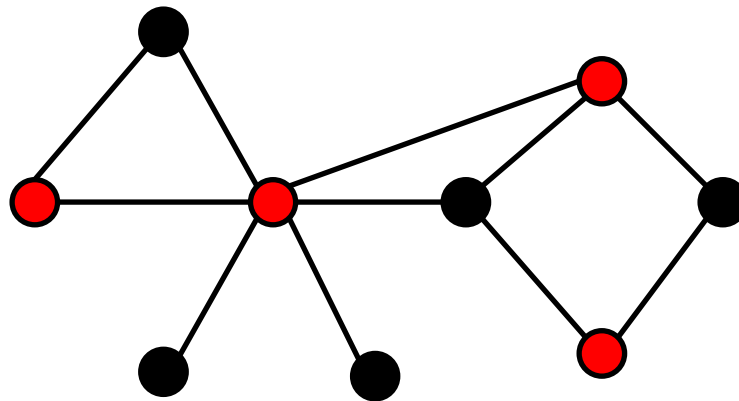
an undirected graph $G=(V,E)$

Feasible solution:

$U \subseteq V$ such that every edge $(u,v) \in E$ is covered, i.e. $u \in U$ or $v \in U$

measure (min):

cardinality of U



a better vertex cover of size 4

Def.

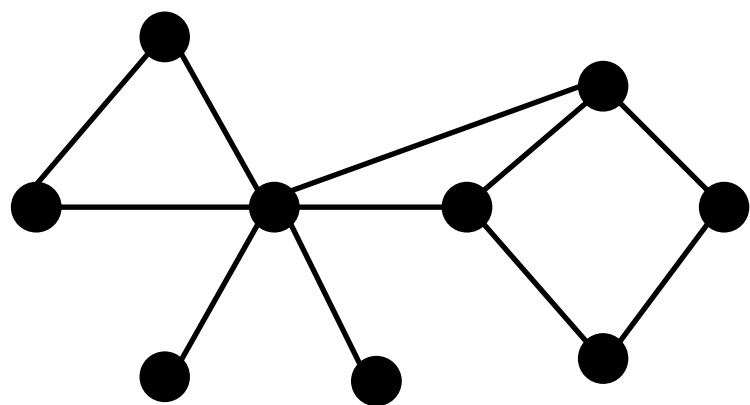
Given a graph $G=(V,E)$, a subset of edges $M \subseteq E$ is a **matching** if no two edges in M share an endpoint.

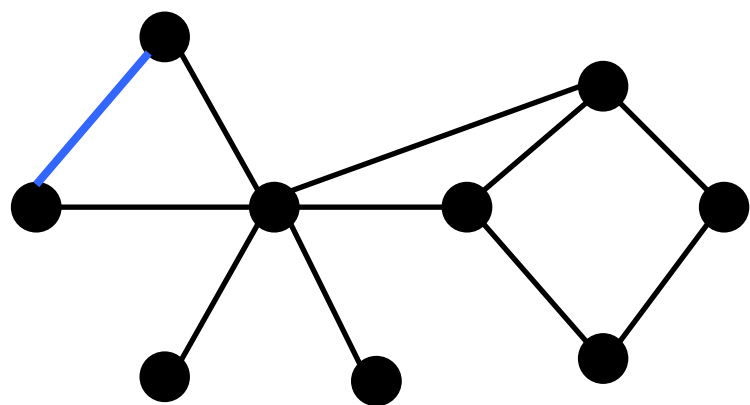
Def.

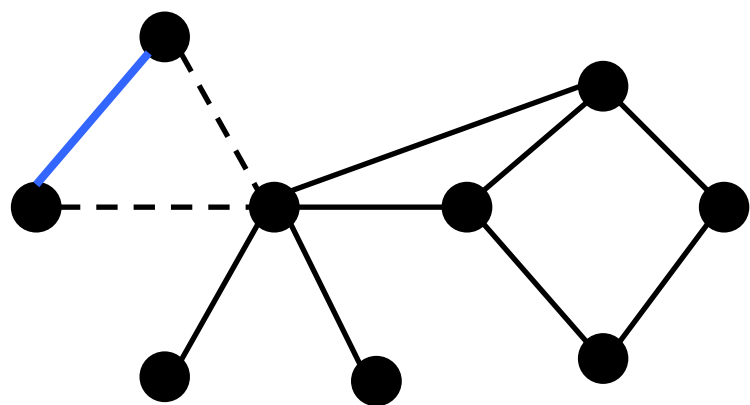
A matching $M \subseteq E$ is **maximal** if for every $e \in E \setminus M$, $M \cup \{e\}$ is not a matching.

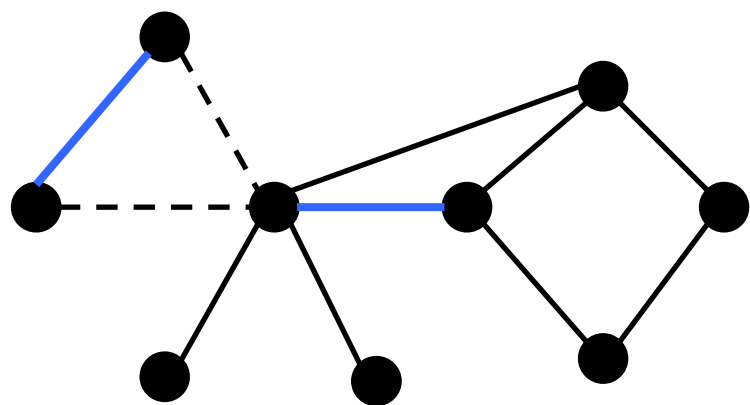
Algorithm 1.2 (Cardinality vertex cover)

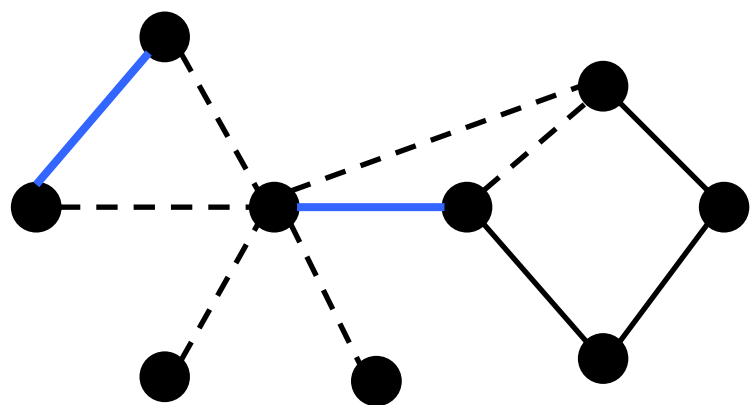
Find a maximal matching in G and output the set of matched vertices.

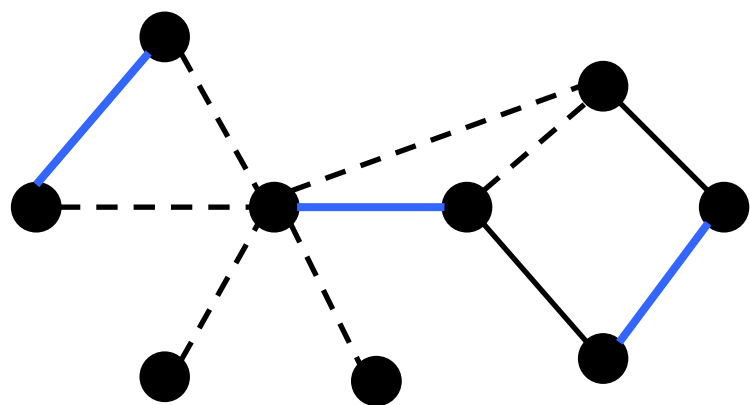


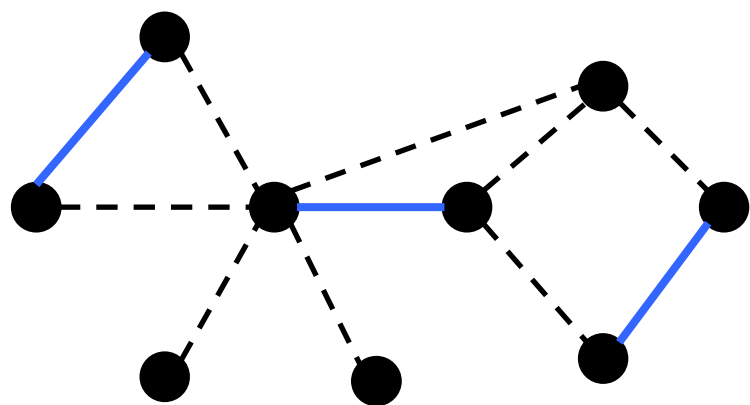


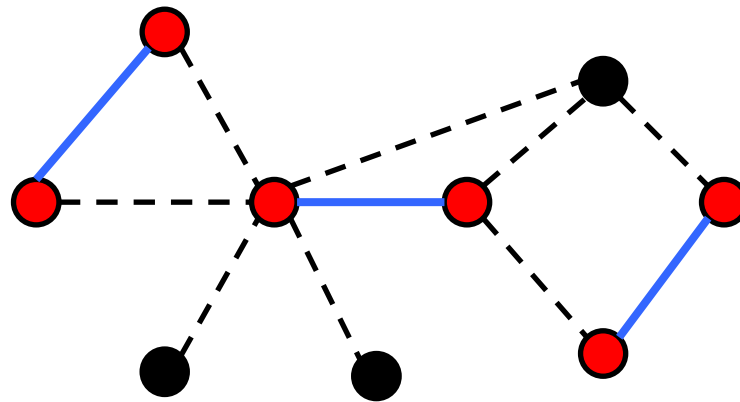












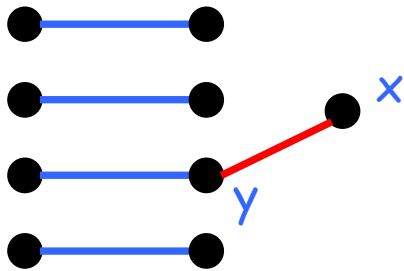
the computed vertex
cover of size 6

Lemma

The algorithm returns a feasible VC.

proof

let $M \subseteq E$ be the maximal matching computed by the algorithm.



edges in M are clearly covered

for maximality of M any other edge (x,y) shares an endpoint with some edge in M ...

...and thus it is covered



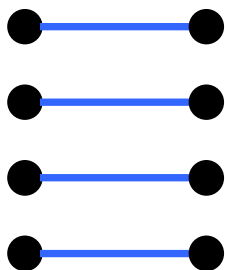
Theorem

The algorithm is a 2-approximation algorithm for the VC problem.

proof

The returned solution is a feasible VC (previous lemma)

let $M \subseteq E$ be the maximal matching computed by the algorithm, and U the corresponding VC.



any optimal solution must have size OPT at least $|M|$

Lower bounding scheme: the size of any maximal matching is a lower bound to the size of an optimal VC

thus:

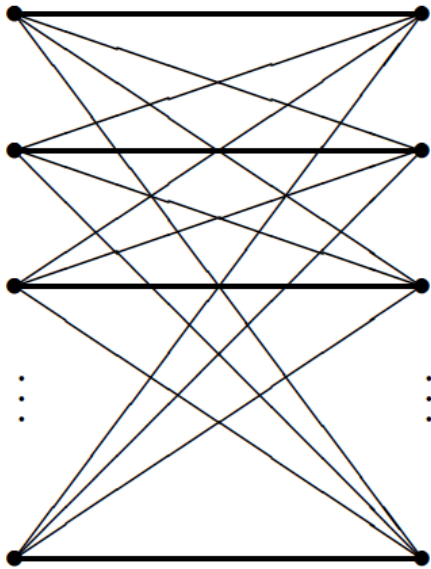
$$|U| = 2|M| \leq 2OPT$$



Three important questions:

1. Can the approximation ratio of Algorithm 1.2 be improved by a better analysis?
2. Can an approximation algorithm with a better apx ratio be designed using the lower bounding scheme of Algorithm 1.2, i.e. the size of a maximal matching?
3. Is there some other lower bounding scheme that can lead to a better approximation algorithm for VC?

1. Can the approximation ratio of Algorithm 1.2 be improved by a better analysis? **NO**



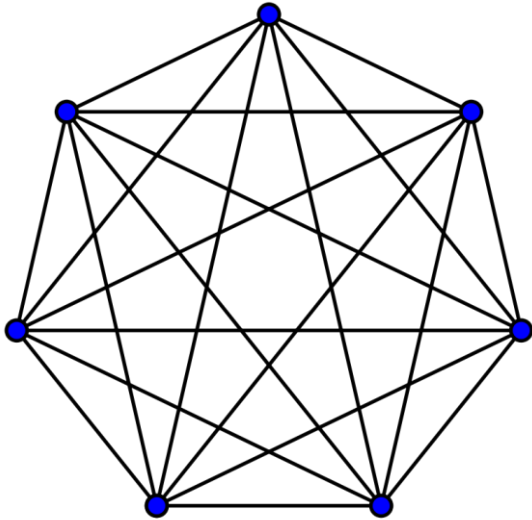
tight example

complete bipartite graph $K_{n,n}$

Algorithm 1.2 will pick all the $2n$ vertices

$\text{OPT} = n$ (one side is an optimal VC)

2. Can an approximation algorithm with a better apx ratio be designed using the lower bounding scheme of Algorithm 1.2, i.e. the size of a maximal matching? **NO**



complete graph K_n where n is odd

size of any maximal matching is $(n-1)/2$

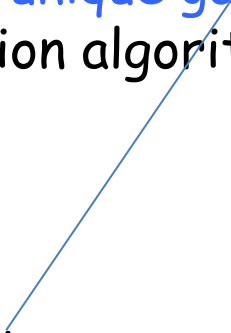
$OPT = n-1$

3. Is there some other lower bounding scheme that can lead to a better approximation algorithm for VC? **OPEN**

Partial answer:

Theorem

Assuming the **unique games conjecture** holds, if there exists an α -approximation algorithm for the VC problem with $\alpha < 2$, then $P=NP$.



roughly: a particular problem (called unique games) is NP-hard

Minimum Set Cover problem

minimum Set Cover problem

Input:

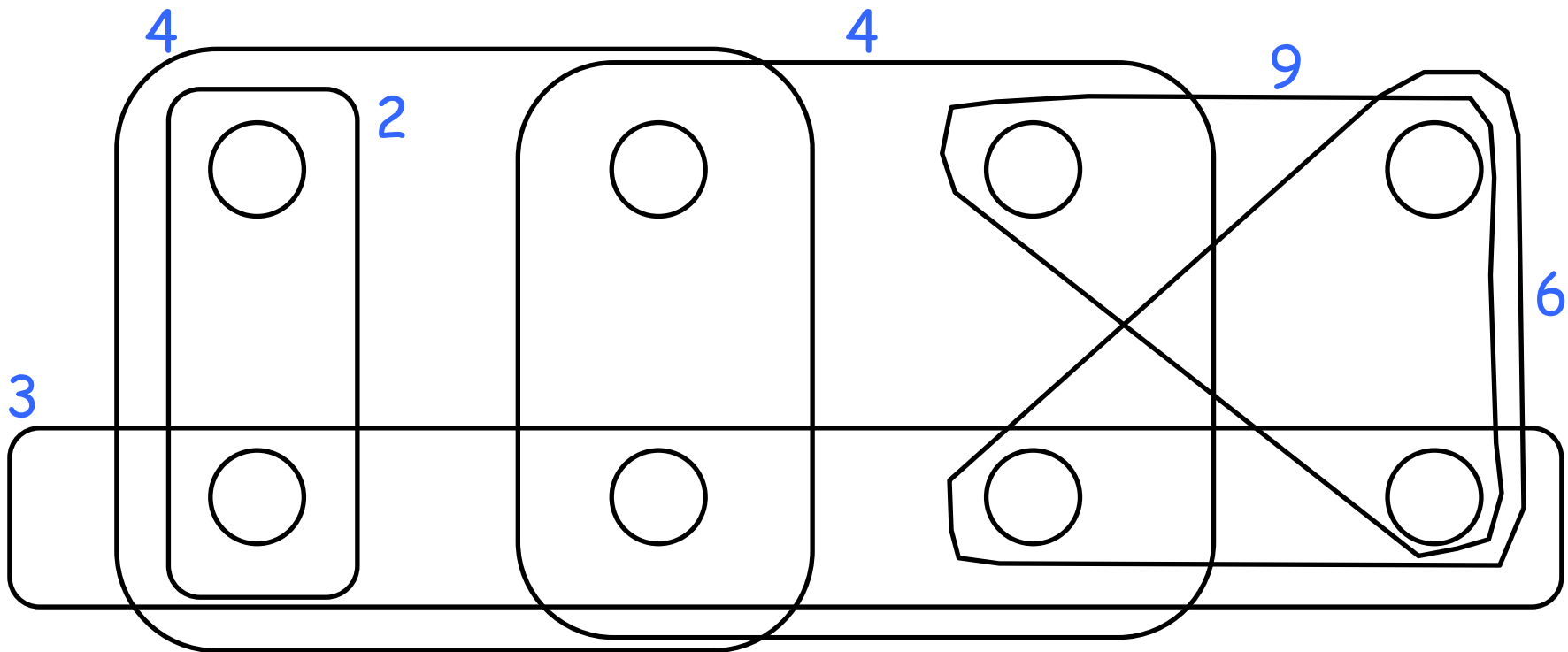
- universe U of n elements
- a collection of subsets of U , $\mathcal{S} = \{S_1, \dots, S_k\}$
- each $S \in \mathcal{S}$ has a positive cost $c(S)$

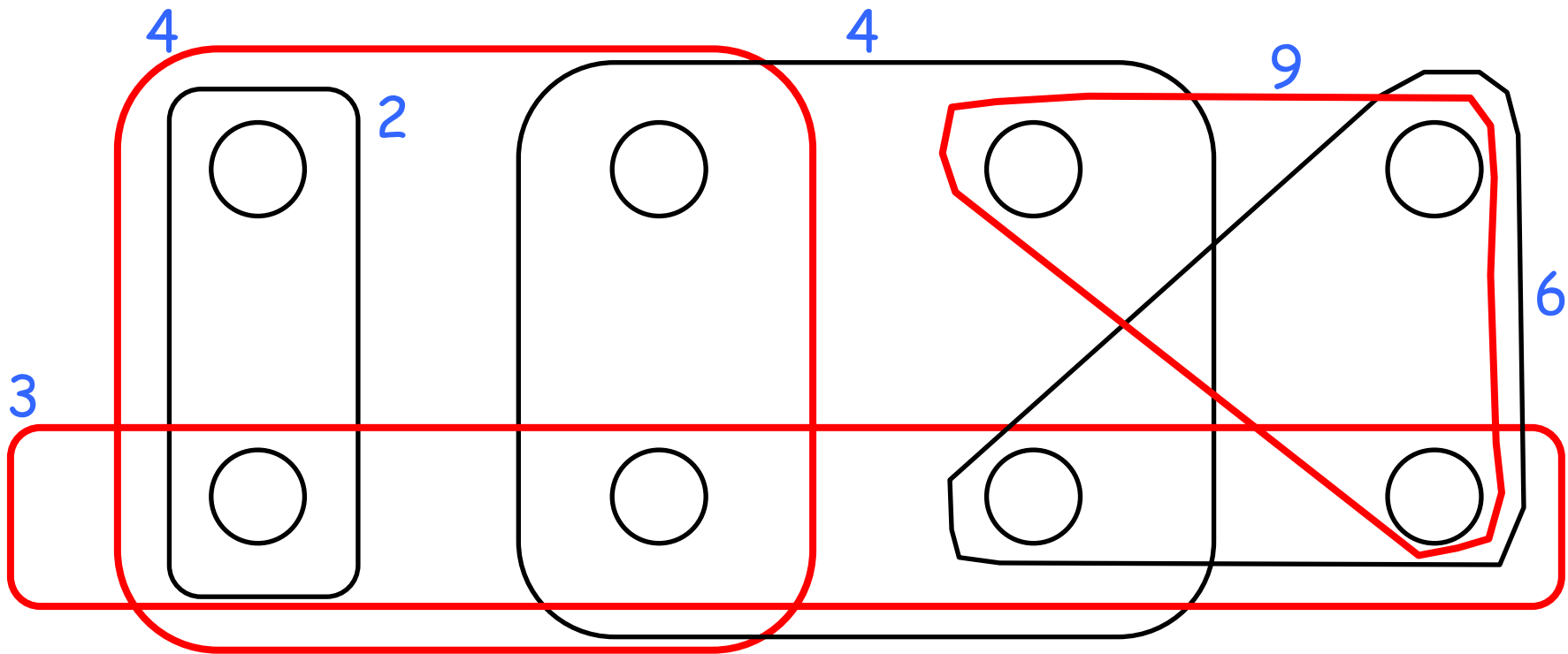
Feasible solution:

a subcollection $\mathcal{C} \subseteq \mathcal{S}$ that covers U (whose union is U)

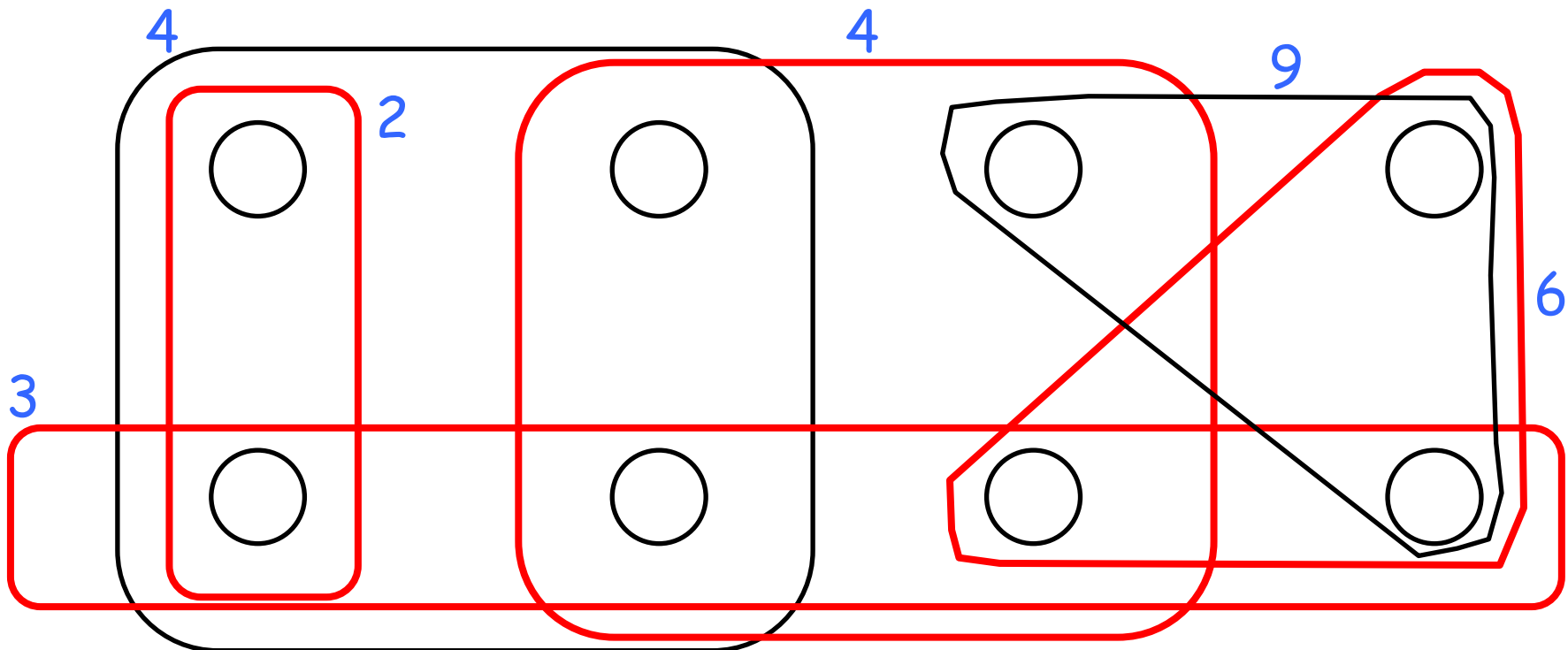
measure (min):

cost of \mathcal{C} : $\sum_{S \in \mathcal{C}} c(S)$

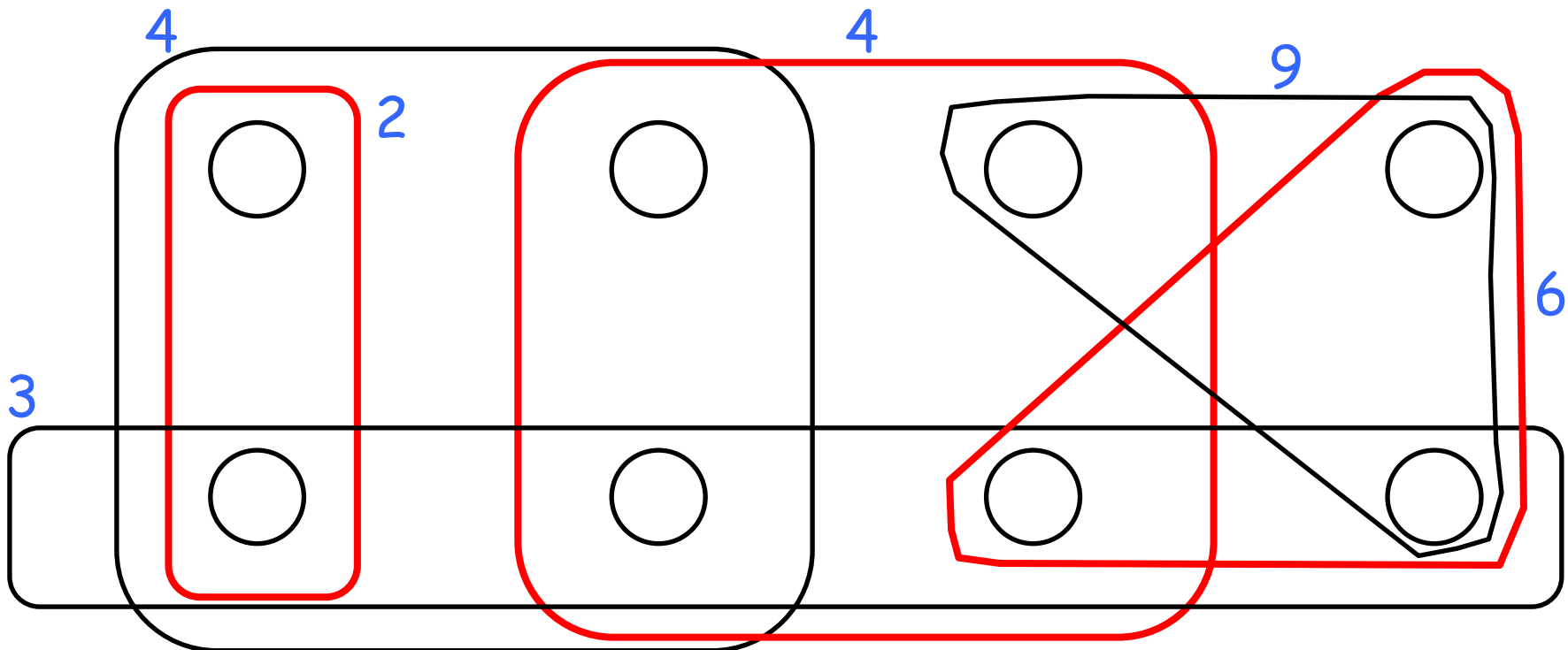




a set cover of
cost 16



a better set
cover of cost 15



a better set
cover of cost 12

greedy strategy: pick the most cost-effective set and remove the covered elements, until all elements are covered.

Let C be the set of elements already covered.

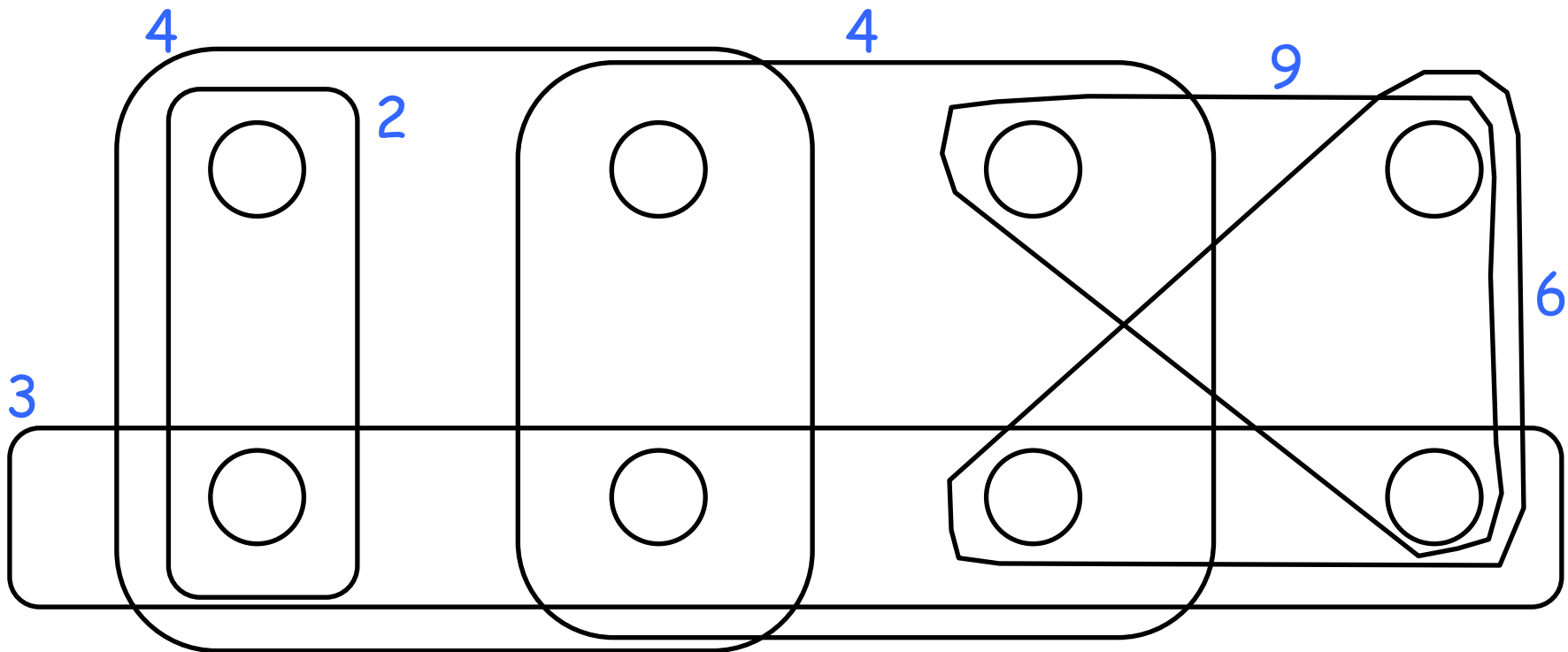
cost-effectiveness of S : $c(S)/|S-C|$

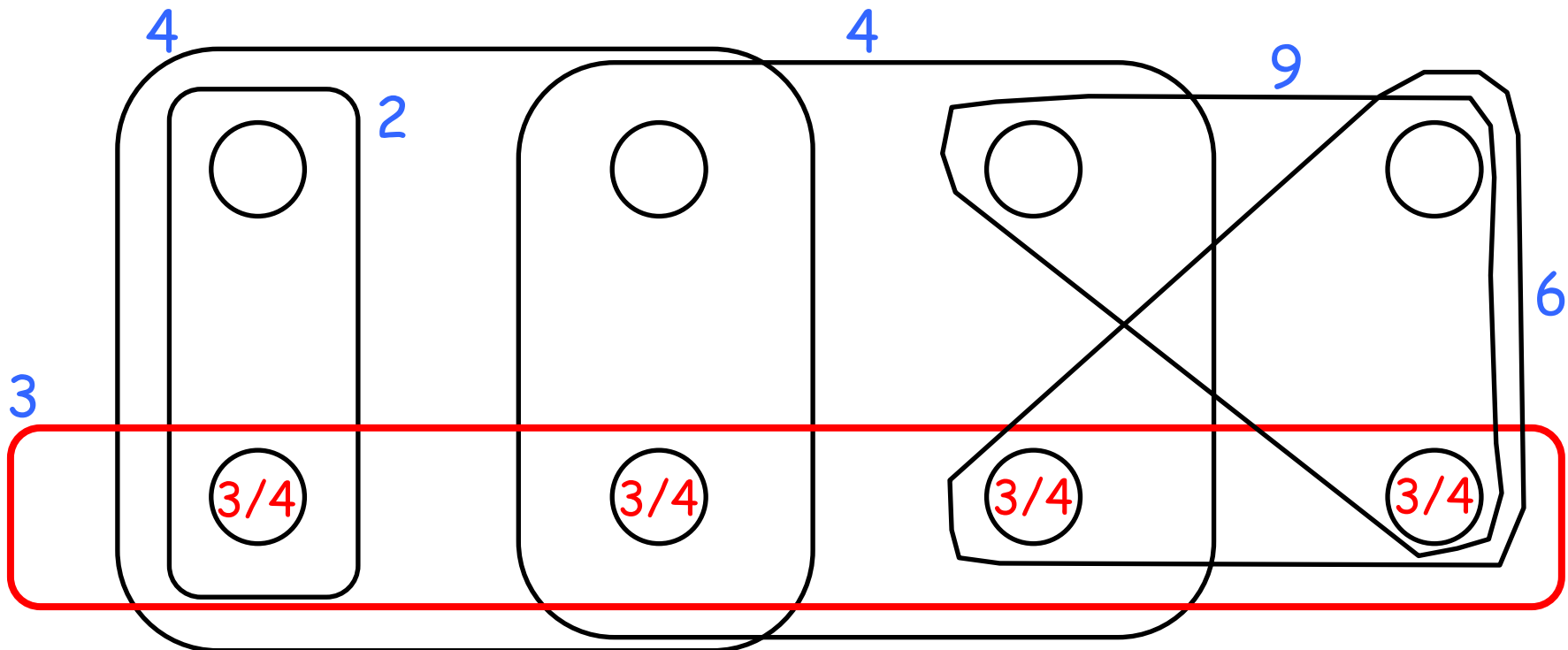
average cost at which
 S covers new elements

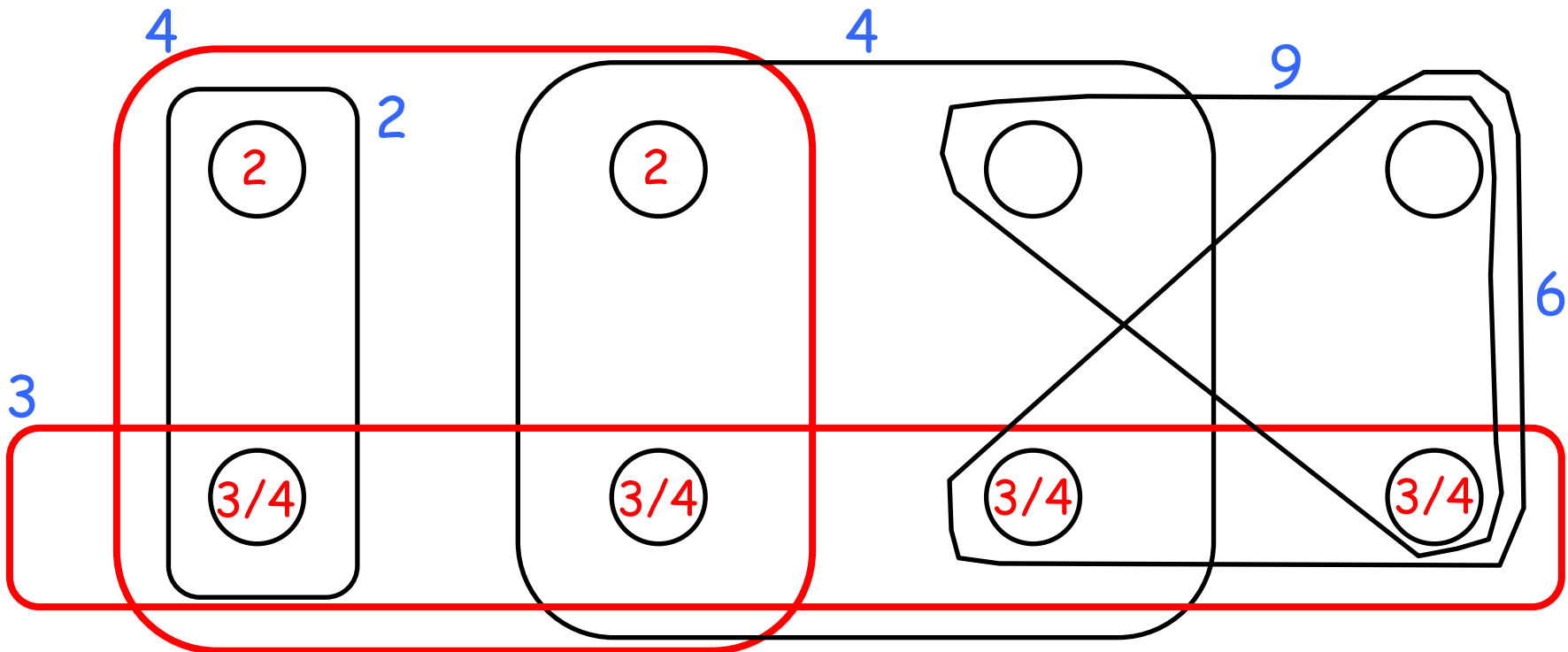
Algorithm 2.2 (Greedy set cover algorithm)

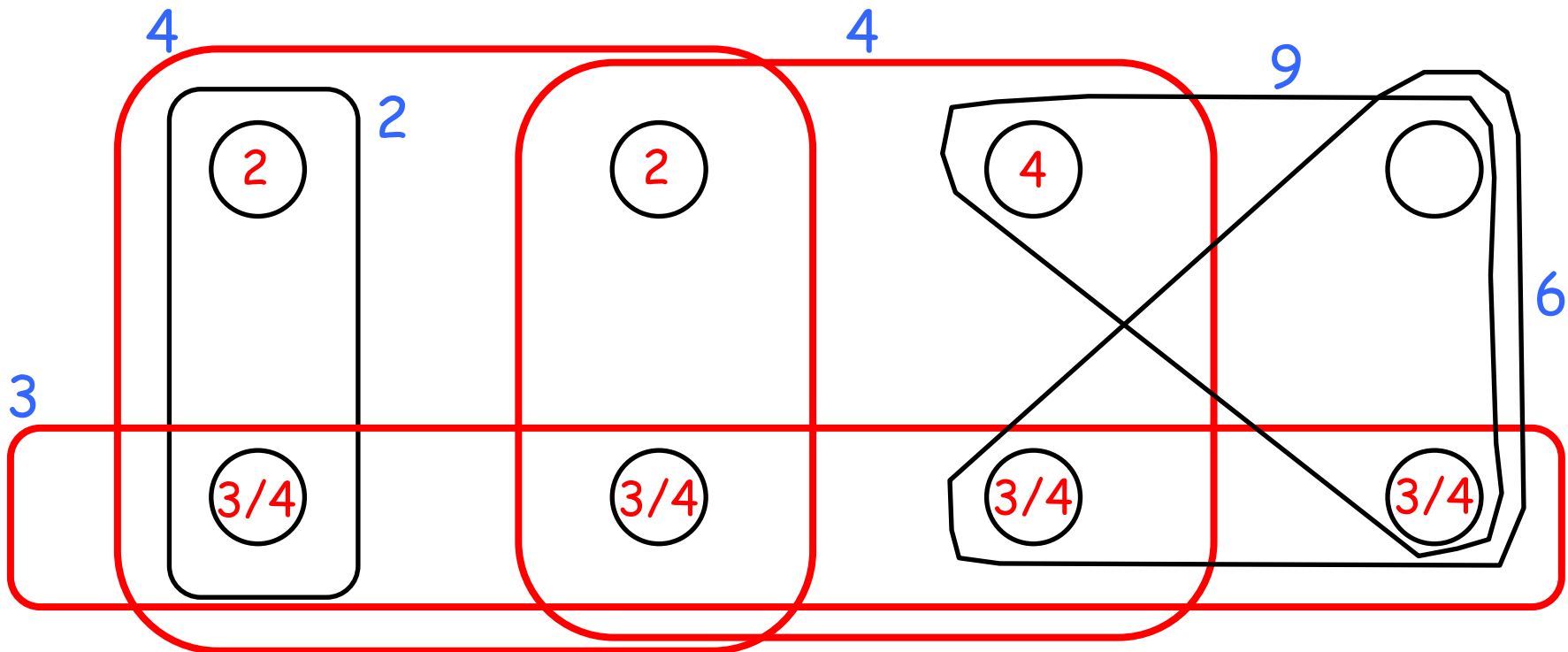
1. $C \leftarrow \emptyset$
2. While $C \neq U$ do
 - Find the most cost-effective set in the current iteration, say S .
 - Let $\alpha = \frac{\text{cost}(S)}{|S-C|}$, i.e., the cost-effectiveness of S .
 - Pick S , and for each $e \in S - C$, set $\text{price}(e) = \alpha$.
 - $C \leftarrow C \cup S$.
3. Output the picked sets.

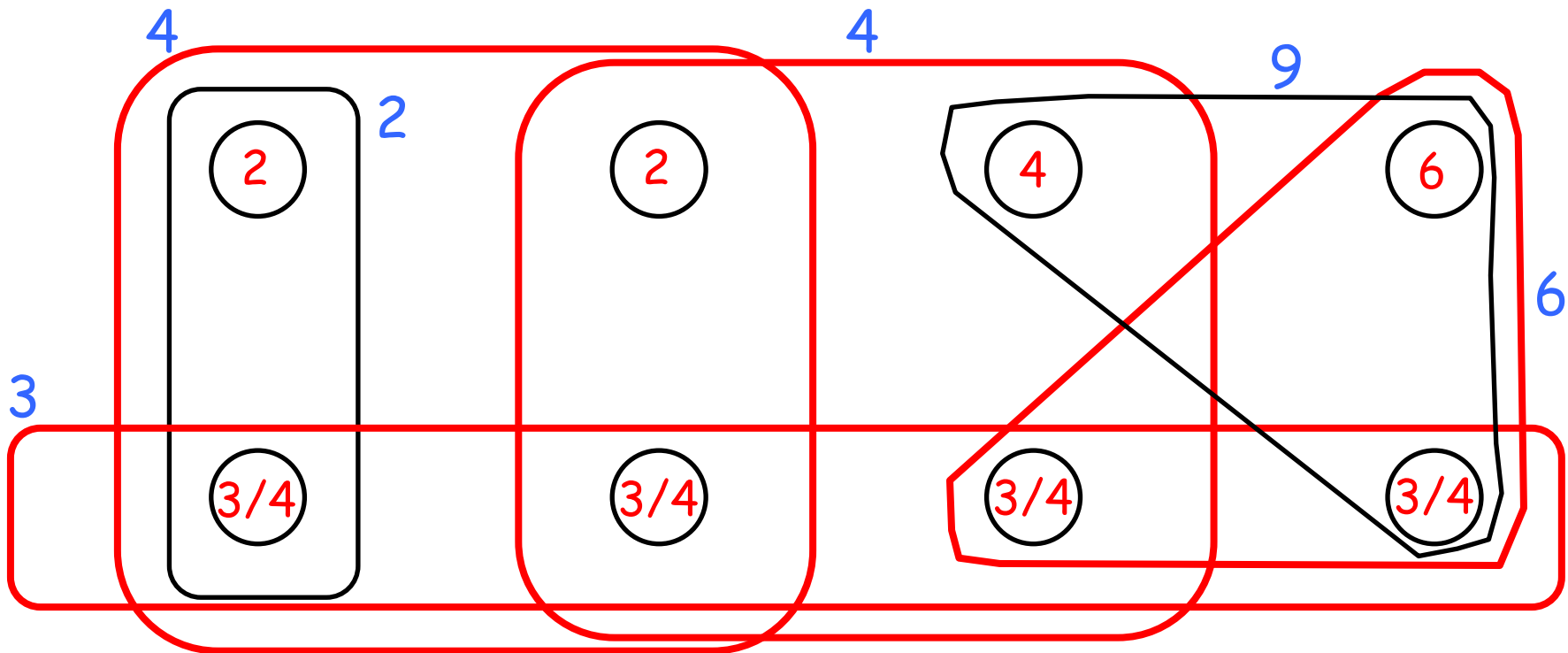
average cost at which
 e is covered











the computed set
cover of cost 17

Number the elements of U in order in which they were covered, resolving ties arbitrarily. Let e_1, \dots, e_n this numbering.

Lemma

For each $k \in \{1, \dots, n\}$, $\text{price}(e_k) \leq \text{OPT}/(n-k+1)$

proof

at any iteration, the leftovers sets of the optimal solution can cover all the remaining elements $C' = U - C$ at cost at most OPT .

one of these leftovers sets has cost-effectiveness at most $\text{OPT}/|C'|$

at iteration in which e_k is covered, C' contains at least $n-k+1$ elements.

by the greedy choice:

$$\text{price}(e_k) \leq \text{OPT}/|C'| \leq \text{OPT}/(n-k+1)$$



Theorem

The greedy algorithm is H_n factor approximation algorithm for the minimum Set Cover problem, where $H_n = 1 + 1/2 + \dots + 1/n$.

proof

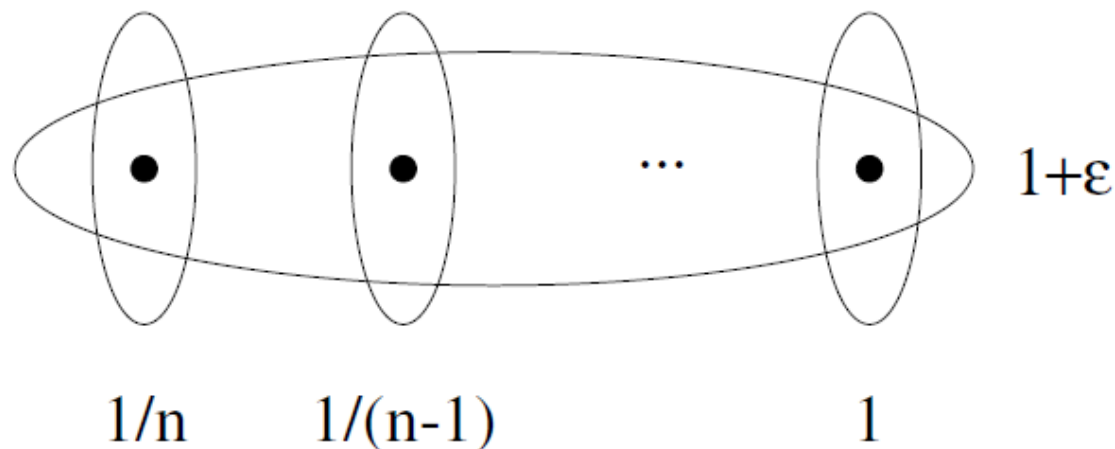
Since the cost of each picked set is distributed among the new covered elements:

$$\text{cost of the cover} = \sum_{k=1}^n \text{price}(e_k) \leq \sum_{k=1}^n \text{OPT} / (n - k + 1) \leq H_n \text{OPT}$$



$$H_n = \sum_{k=1}^n 1/k \leq \ln n + 1 \quad \text{n-th harmonic number}$$

tight example



the greedy alg
computes a cover having
cost H_n

$$\text{OPT} = 1 + \varepsilon$$

Theorem

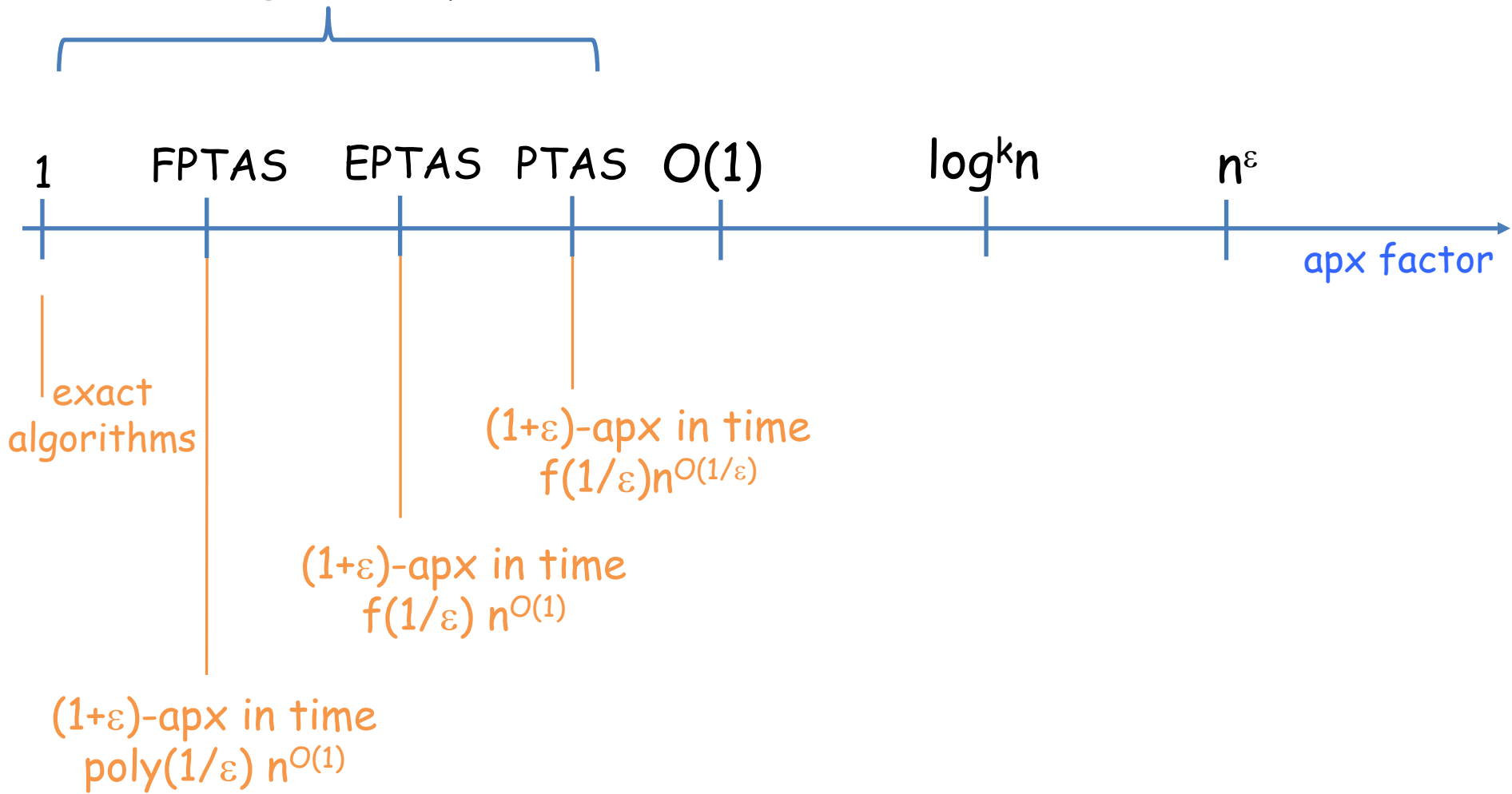
There exists some constant $c > 0$ such that if there exists a $(c \ln n)$ -apx algorithm for the unweighted SC problem, then $P = NP$.

Theorem

If there exists a $(c \ln n)$ -apx algorithm for the unweighted SC problem, for some constant $c < 1$, then there is an $O(n^{O(\log \log n)})$ -time alg for each NP-complete problem.

the approximation game: get better and better approximation factor

Polynomial-Time Approximation Scheme:
(1+ ϵ)-apx for any $\epsilon > 0$.
running time depends on ϵ



Application to shortest superstring

the shortest superstring problem

Input:

a set of n strings over a finite alphabet $S=\{s_1, \dots, s_n\}$

Feasible solution:

a string s that contains each s_i as a substring

measure (min):

length of s

notice: w.l.o.g. we can assume no string s_i is a substring of another s_j

$S=\{abbc, cccaab, bccc\}$

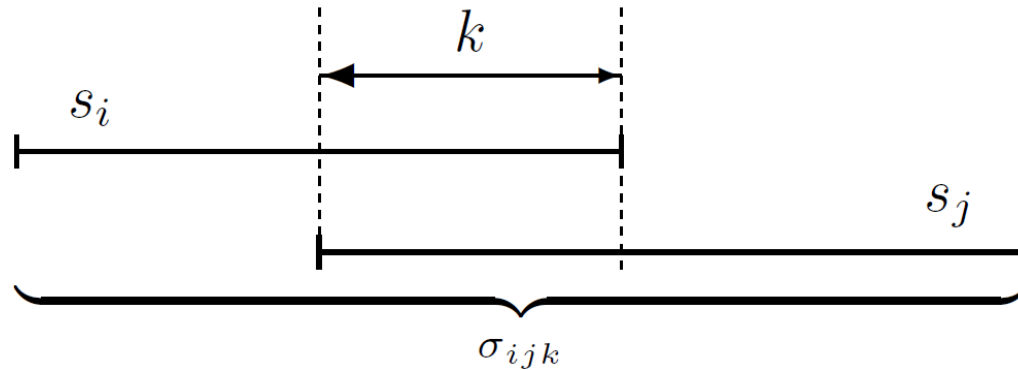
a solution of length 12: abbcccaabccc

a better solution of length 9: bcccaabbc

reducing the problem to set cover

for $s_i, s_j \in S$, and $k > 0$

if the last k symbols of s_i are the same as the first k symbols of s_j ,
let σ_{ijk} be the string obtained by overlapping those k positions



let M be the set of the strings σ_{ijk} for all valid choices of i, j, k .

for a given string π ,

let $\text{set}(\pi) = \{s \in S : s \text{ is a substring of } \pi\}$

the Set Cover instance:

- the set of objects is S
- collection of subsets: we have $\text{set}(\pi)$ for each $\pi \in S \cup M$ of cost $|\pi|$

Algorithm 2.10 (Shortest superstring via set cover)

1. Use the greedy set cover algorithm to find a cover for the instance \mathcal{S} .
Let $\text{set}(\pi_1), \dots, \text{set}(\pi_k)$ be the sets picked by this cover.
2. Concatenate the strings π_1, \dots, π_k , in any order.
3. Output the resulting string, say s .

Theorem

The above algorithm is a $2H_n$ -approximation algorithm for the shortest superstring problem.

proof

since we have computed a set cover, every $s \in S$ is a substring of some π_j

➡ the computed string is a feasible superstring

OPT : the value of the optimal solution for the shortest superstring

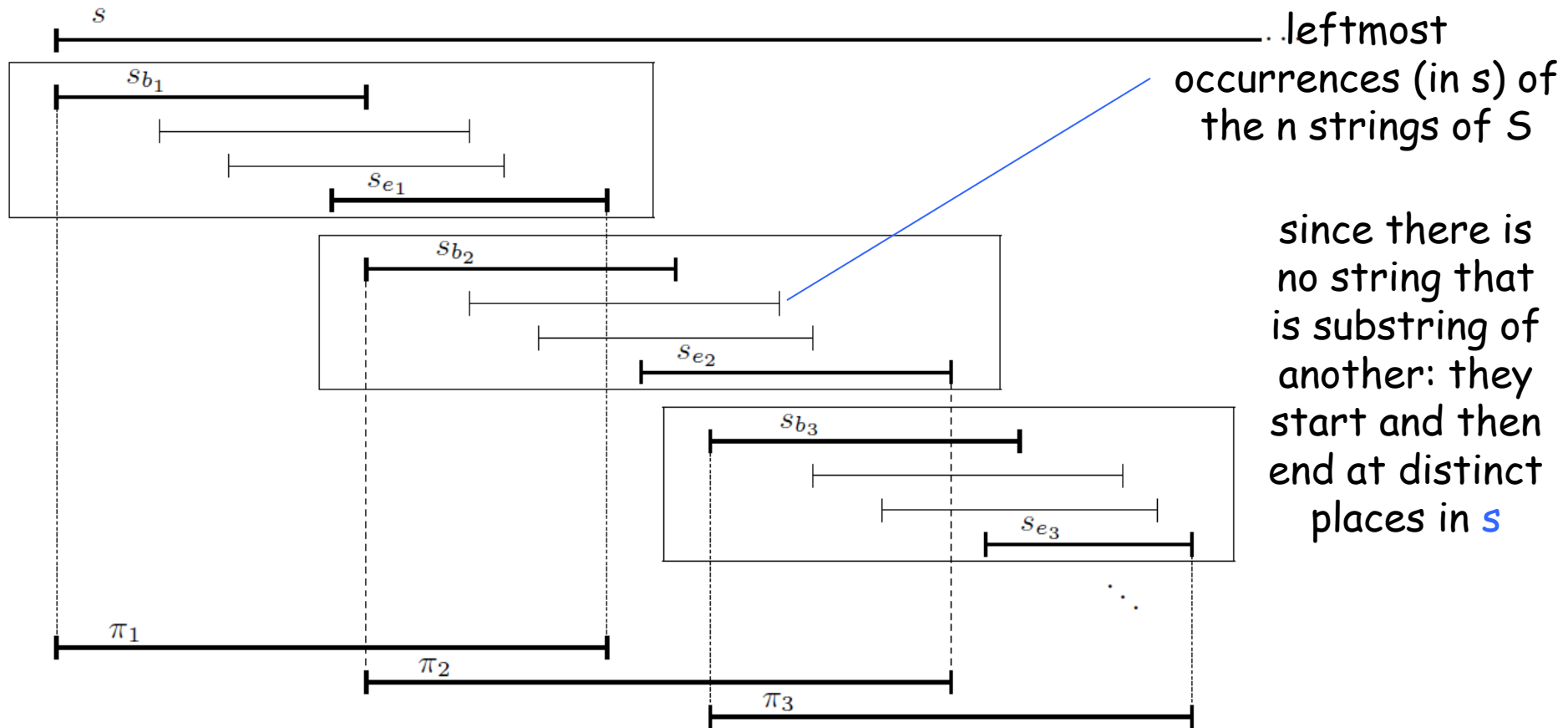
OPT_{SC} : the value of the optimal solution for the SC instance

claim: $\text{OPT}_{\text{SC}} \leq 2 \text{ OPT}$.

➡ the computed string has length $\leq H_n \text{OPT}_{\text{SC}} \leq 2H_n \text{OPT}$

Let s be the optimal superstring of length OPT

we show there is a feasible SC of cost at most $2 OPT$



all the sets $\text{set}(\pi_i)$ is a feasible SC of cost $\sum_i |\pi_i|$

notice: π_i and π_{i+2} do not overlap



$$\sum_i |\pi_i| \leq 2 |s| \leq 2 OPT$$

