Introduction to
Approximation Algorithms

Mong-Jen Kao (高孟駿)

Friday 13:20 – 15:10

Outline

- The Set Cover Problem
  - An H_n-approximation via greedy approach
    - The cost-efficiency of the choices
    - A tight example for the algorithm analysis
  - An $O(\log n)$-Approximation via randomized LP-rounding

The Set Cover Problem

The Set Cover Problem

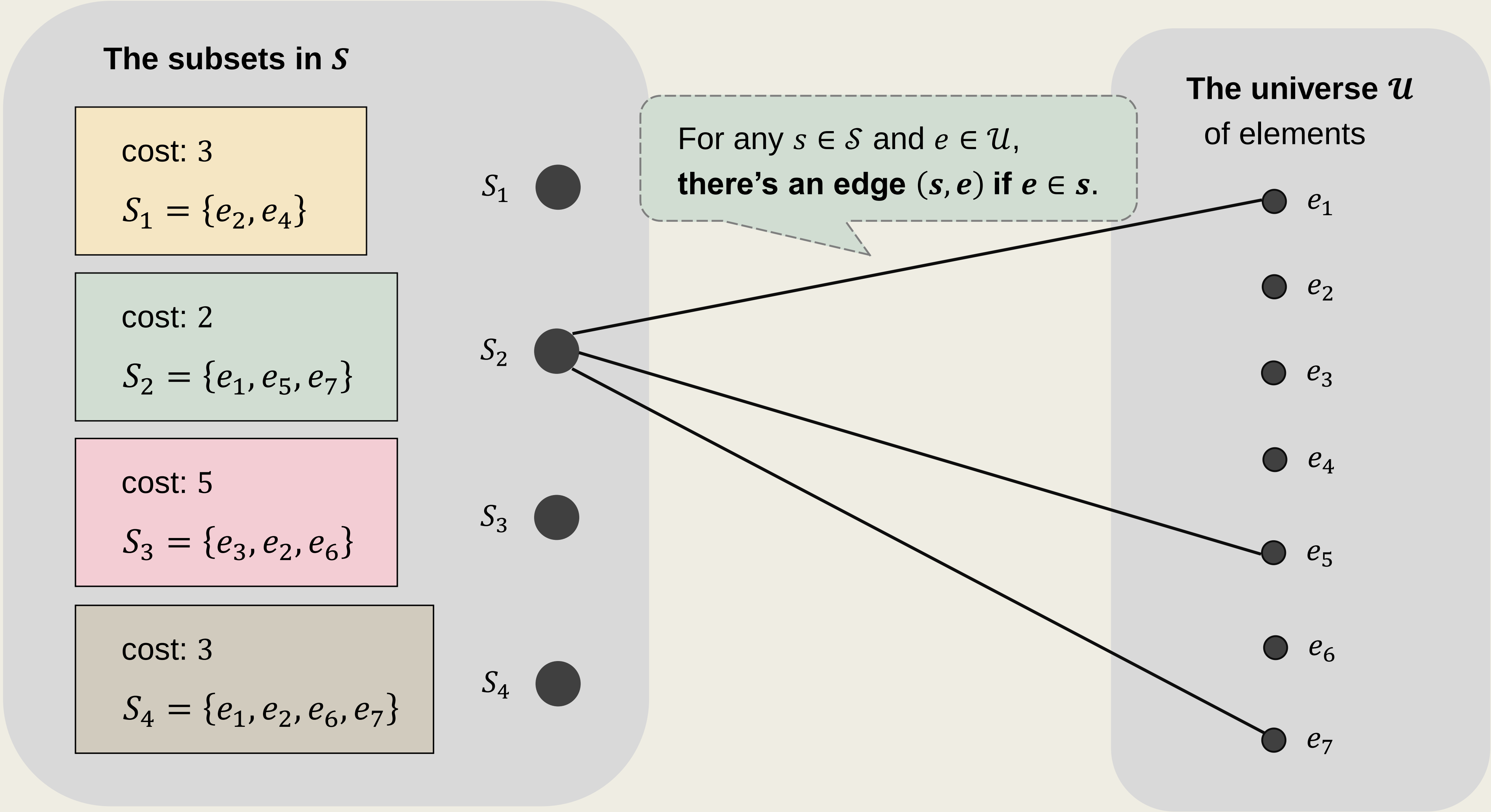
- Given *a universe $\mathcal{U}$ of n elements*, a collection of subsets of $\mathcal{U}$, $\mathcal{S} = \{ S_1, S_2, \ldots, S_k \}$, and a cost function $c : \mathcal{S} \rightarrow \mathbb{Q}^+$,

  the set cover problem is to *compute a minimum cost subcollection of $\mathcal{S}$* that covers all the elements of $\mathcal{U}$.

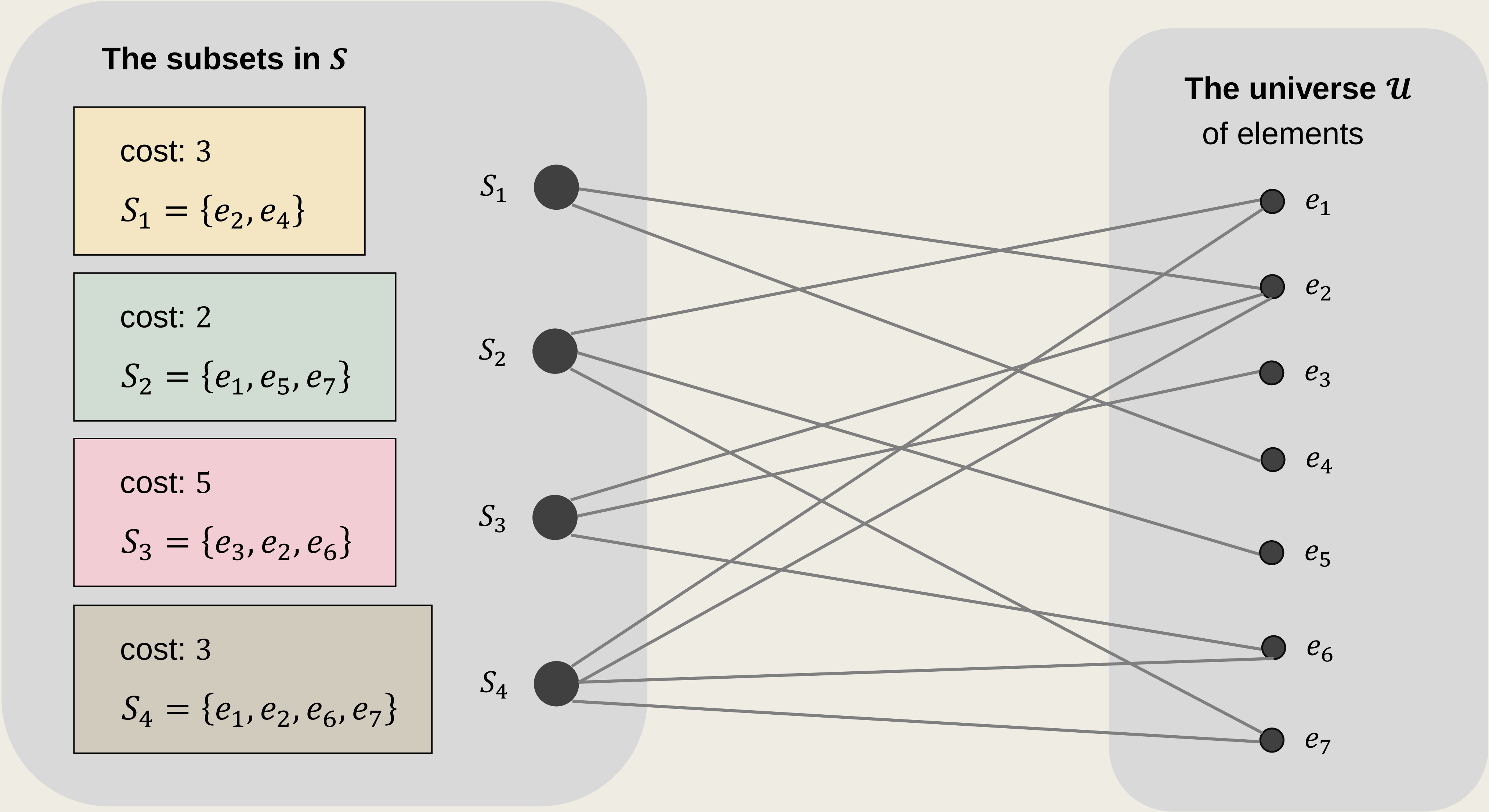
  – i.e., *to pick a collection of subsets $\mathcal{A} \subseteq \mathcal{S}$* such that

    $\bigcup_{s \in \mathcal{A}} s = \mathcal{U}$ and ***the total cost, $\sum_{s \in \mathcal{A}} c(s)$, is minimized***.

An Intuitive Way to View the Set Cover Problem



Pick a minimum cost vertex subset from the left, such that *every vertex on the right* is adjacent to *at least one chosen vertex on the left*.



Pick a minimum cost vertex subset from the left, such that *every vertex on the right* is adjacent to *at least one chosen vertex on the left*.

The subsets in S

cost: 3

$S_1 = \{e_2, e_4\}$

cost: 2

$S_2 = \{e_1, e_5, e_7\}$

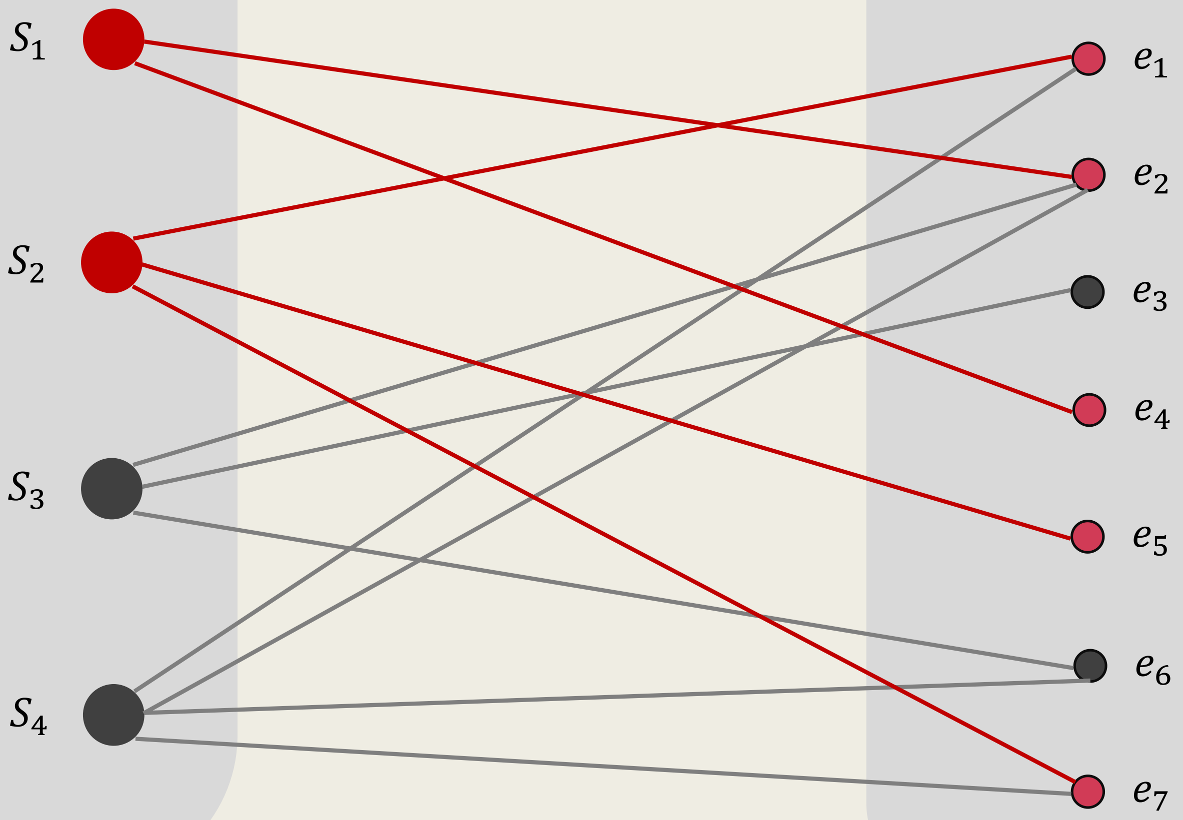
cost: 5

$S_3 = \{e_3, e_2, e_6\}$

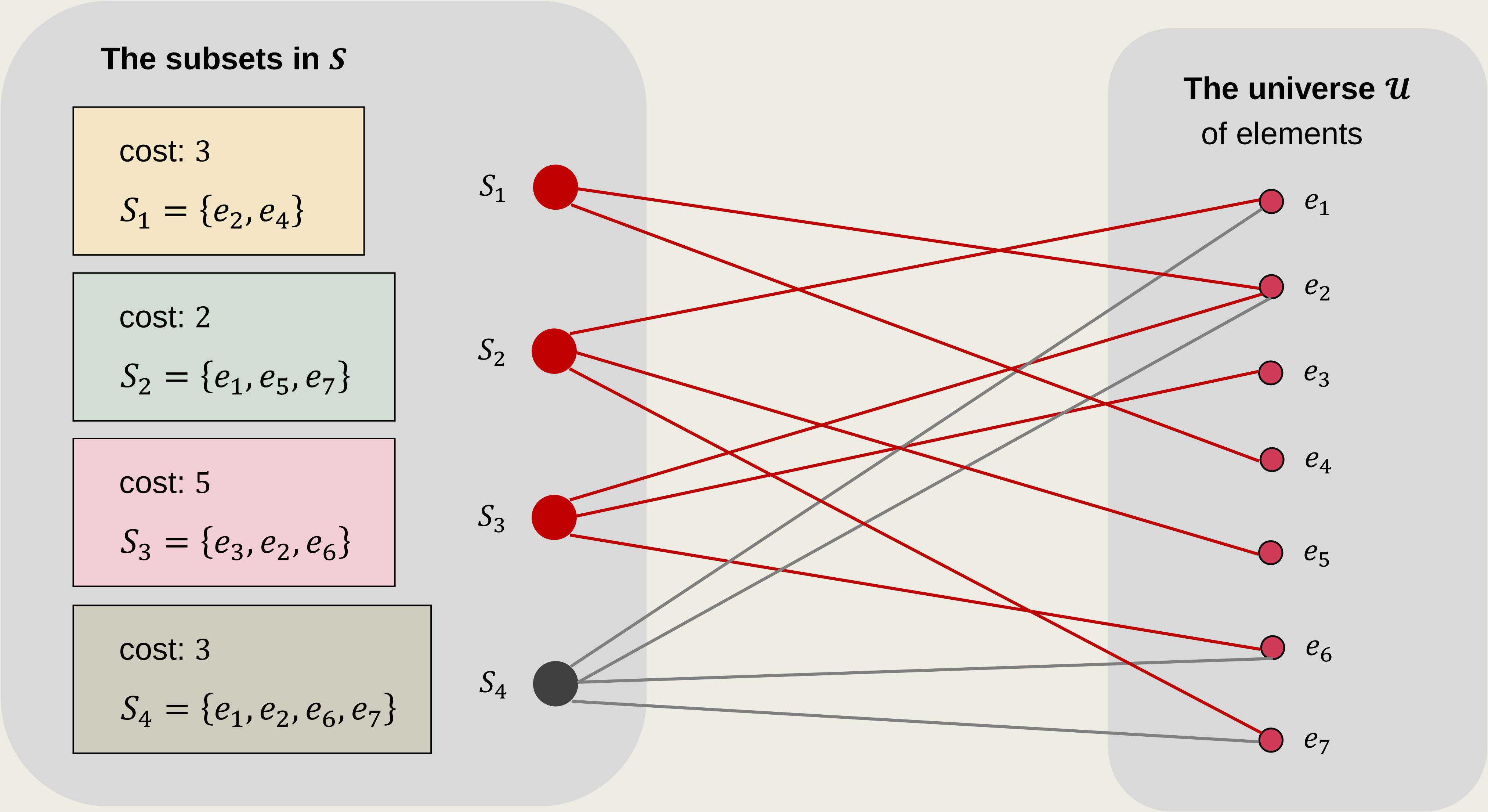
cost: 3

$S_4 = \{e_1, e_2, e_6, e_7\}$

The universe $\mathcal{U}$ of elements



Pick a minimum cost vertex subset from the left, such that *every vertex on the right* is adjacent to *at least one chosen vertex on the left*.



Common Parameters for Set Cover

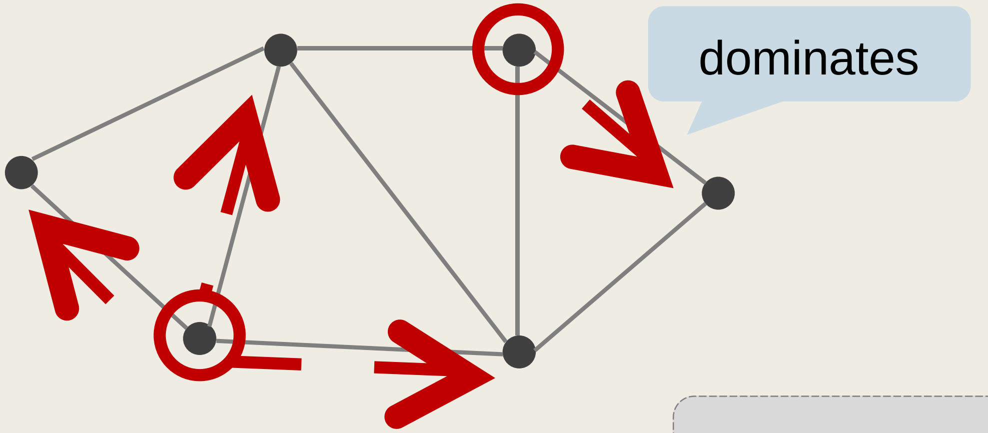
- Let $\Pi = (\mathcal{U}, \mathcal{S}, c)$ be an instance of the set cover problem.
  - For each $u \in \mathcal{U}$, we define the frequency of u to be the number of sets in $\mathcal{S}$ to which u belongs, i.e., the number of sets u is in.
  - We will use f to denote ***the maximum frequency of the elements***.
    - It turns out that, *the maximum frequency is a useful parameter* when approximating the set cover problem.

Related Variations

The Dominating Set Problem

- Given a graph $G = (V, E)$ and a vertex weight function $w : V \to Q^+$, compute a minimum-weight vertex subset $U \subseteq V$ such that, for any $v \in V$, either

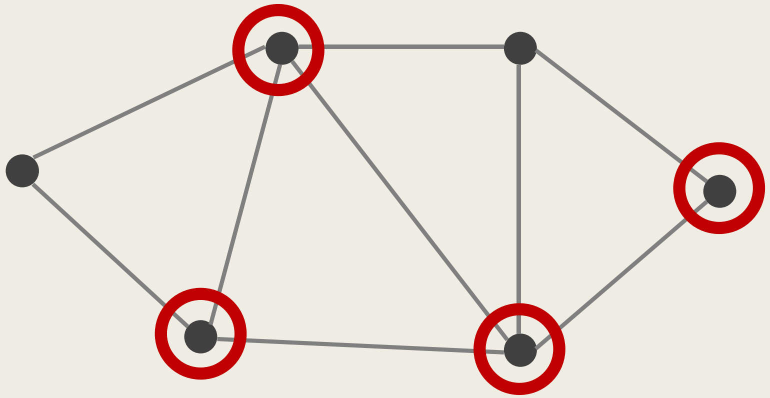
  $v \in U$ or v has a neighbor that does.



Intuitively, we are covering the vertices using the vertices.

The Vertex Cover Problem

- Given a graph $G = (V, E)$ and a vertex weight function $w : V \to Q^+$, compute a minimum-weight vertex subset $U \subseteq V$ such that, for any edge $e \in E$, at least one endpoint of e is in U.
  - The vertex cover problem is a special case of set cover for which $f = 2$.
  - When hypergraphs are considered, vertex cover is equivalent to set cover.



Intuitively, we are covering the edges using the vertices.

(Brief)

Status of the Set Cover Problem

The Set Cover Problem

- The set cover problem is a classic NP-hard problem that is studied in many fields.

- The set cover problem can be approximated to a ratio of
  - H_n by simple greedy approach, where H_n is the n^{th}-harmonic number.
  - f by the "layering" algorithm, where f is the maximum frequency of the elements.

The Set Cover Problem

- The set cover is NP-hard to approximate to $\left(1 - o(1)\right) \cdot \ln n$ unless P=NP.

- If we assume the Unique Game Conjecture (UGC), then approximating set cover to a ratio better than $f - \epsilon$ for any $\epsilon > 0$ is NP-hard.

H_n-approximation by

Simple Greedy Approach on Cost-Efficiency

Greedy towards Cost-Efficiency

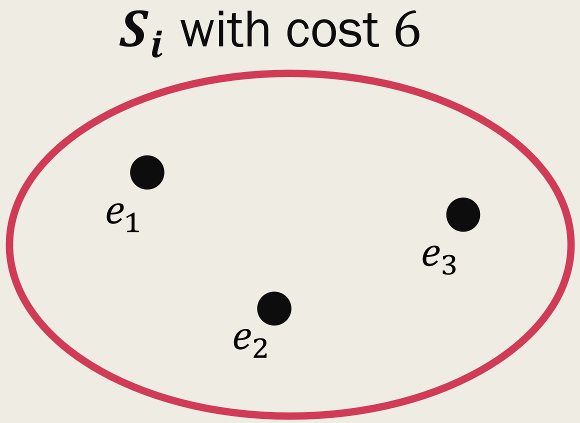
- For problems of this kind, a very natural approach is to consider the ***cost-effectiveness*** / ***cost-efficiency*** of the choices, and to *always* pick the *most cost-efficient one*.
  - This is likely fail for most of the times, if our goal is to solve the optimization problem for an optimal solution.
    - For example, this can perform arbitrarily bad for the knapsack problem.
  - However, this intuitive approach yields a good approximation for the set cover problem, *provably* *the best one*.

How is Cost-Efficiency Defined?

- One natural question is that,

  How should the ***cost-efficiency*** of the sets be defined?

  – It may seem that…



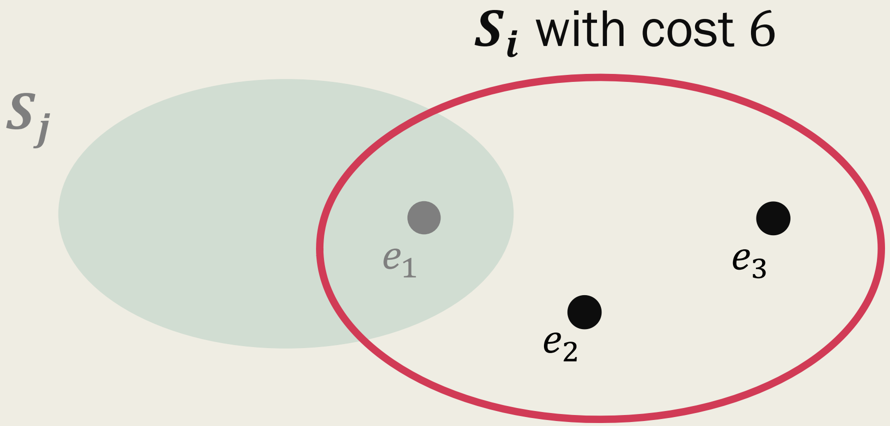
Selecting S_i can cover 3 elements with a cost of 6.

The ***average price*** of S_i is $6/3 = 2$.

This may seem correct, but…

How is Cost-Efficiency Defined?

- The ***cost-efficiency*** of the sets can change as the algorithm proceeds.
  - Suppose that, prior to picking S_i, some sets were already picked…



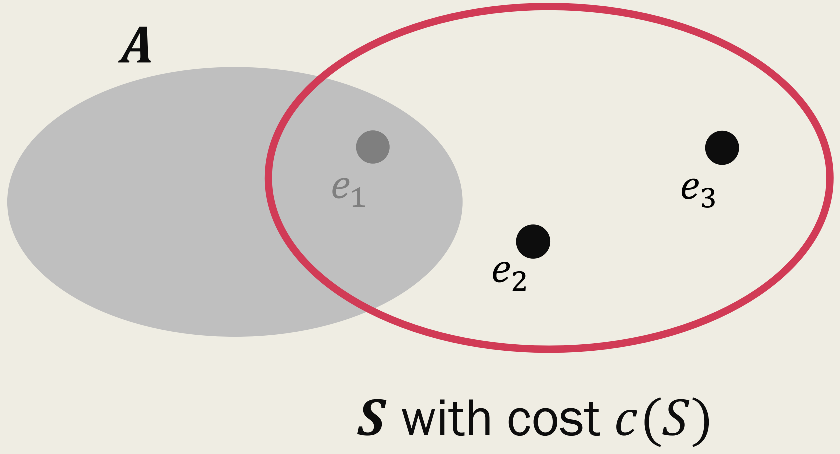
Selecting S_i can cover only 2 elements.

The ***average price*** of S_i is now $6/2 = 3$, instead of 2.

How is Cost-Efficiency Defined?

- Let A be the set of elements that have already been covered.
  - We define the average covering price of a set S, subject to a prior coverage of A, to be

$$\text{Aprice}(S, A) := \frac{c(S)}{|S - A|}.$$



The Algorithm Description

The algorithm

- The algorithm **picks the *most cost-efficient subset*** in each iteration *until all the elements are covered*.

> • While C is not yet a cover,
>
> Pick the most cost-efficient subset from $\mathcal{S}$ and add it to C.

  – The idea is that,
    since we always pick the *“best choice” in each iteration*,
    its efficiency is ***no worse than that of the optimal solution***.

The algorithm

- The algorithm **picks the *most cost-efficient subset*** in each iteration *until all the elements are covered*.

$\mathcal{C} \leftarrow \emptyset$.

while $\bigcup_{s \in \mathcal{C}} \boldsymbol{s} \neq \mathcal{U}$, **do**

Pick the set $S' \in \mathcal{S}$ with the minimum $\mathrm{aprice}(S', \bigcup_{s \in \mathcal{C}} s)$.

$\mathcal{C} \leftarrow \mathcal{C} \cup \{S'\}$.

Return $\mathcal{C}$.

The Analysis

The Approximation Guarantee

- Let $e_1, e_2, \dots, e_n$ be the elements in $\mathcal{U}$, with indexes labelled by the order they are covered.
  - Define $\mathrm{price}(e_i)$ to be the price the algorithm uses to cover e_i, i.e., the average price of the particular set that first makes e_i covered.

- The following lemma, which bounds the covering price of each element, is the key to establishing the H_n guarantee.

Lemma 1.

We have $\mathrm{price}(e_i) \leq \frac{OPT}{n-i+1}$ for all $1 \leq i \leq n$.

The Approximation Guarantee

Lemma 1.

We have $\text{price}(e_i) \leq \frac{OPT}{n-i+1}$ for all $1 \leq i \leq n$.

- Suppose that Lemma 1 is true, then it follows that

$$c(\mathcal{C}) := \sum_{S \in \mathcal{C}} c(S) = \sum_{1 \leq i \leq n} \text{price}(e_i) \leq \sum_{1 \leq i \leq n} \frac{1}{i} \cdot OPT$$

$$= H_n \cdot OPT.$$

The cost of each $S \in \mathcal{C}$ is *distributed* as *the prices of the elements* *it effectively covers*.

So, it suffices to prove Lemma 1.

An intuitive lemma with a technical proof.

Lemma 1.

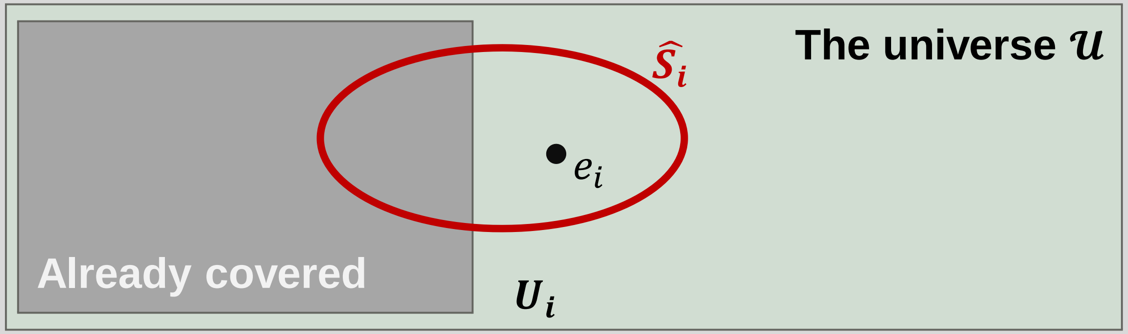
We have $\text{price}(e_i) \leq \frac{OPT}{n-i+1}$ for all $1 \leq i \leq n$.

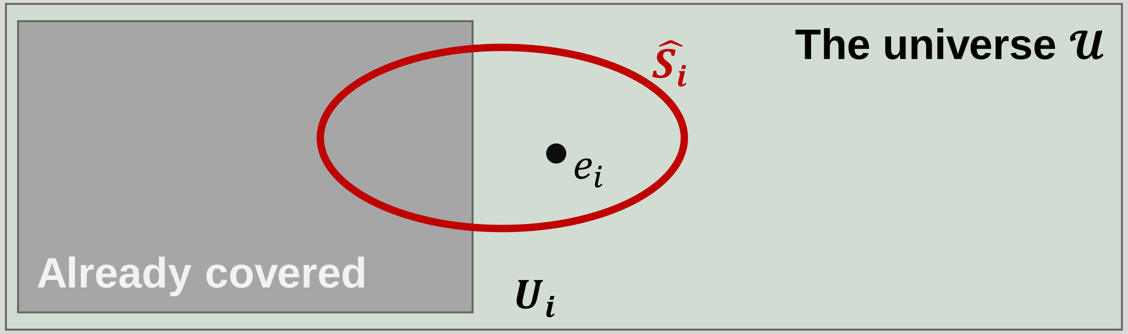
Proof.

Consider ***the particular iteration*** for which $\boldsymbol{e_i}$ ***becomes covered***.

Let $\widehat{S}_{\boldsymbol{i}}$ denote the set that is picked to cover e_i, and

U_i denote set of uncovered elements in the beginning of that iteration.





The optimal solution (for $(\mathcal{U}, \mathcal{S}, c)$) can cover U_i with cost OPT.

Since $\widehat{S}_i$ is *the* ***most cost-efficient choice*** *at that moment*,

we claim that *its average price is at most* $OPT/|U_i|$.

If so, then

$$\text{price}(e_i) \leq \frac{OPT}{|U_i|} \leq \frac{OPT}{n-i+1}.$$

The average price of the optimal solution at that moment.

e_i is the i^{th}-element that gets covered. So, $|U_i| \geq n - i + 1$.

The average price of $\widehat{S}_i$ subject to <u>*prior coverage*</u> of $\mathcal{U} - U_i$.

<u>Proof. (continue)</u>

It remains to prove the claim that $\text{aprice}\left(\widehat{S}_i, \mathcal{U} - U_i\right) \leq \frac{OPT}{|U_i|}$.

Let $\mathcal{O} = \{O_1, O_2, \dots, O_\ell\}$ denote ***an optimal solution for*** $(\boldsymbol{U_i}, \boldsymbol{\mathcal{S}}, \boldsymbol{c})$.

- Imagine that, $O_1, O_2, \dots, O_\ell$ are selected in order.
- For any $1 \leq j \leq \ell$, define

$$\text{ap}'(O_j) := \text{aprice}\left(O_j, (U - U_i) \cup \bigcup_{1 \leq k < j} O_k\right).$$

Intuitively, $\text{ap}'(O_j)$ is the updated average price of O_j, when $O_1, O_2, \dots, O_{j-1}$ are selected in prior to O_j.

Proof. (continue)

It remains to prove the claim that $\text{aprice}(\widehat{S}_i, \mathcal{U} - U_i) \le \frac{OPT}{|U_i|}$.

Denote by $\mathcal{O} = \{O_1, O_2, \dots, O_\ell\}$ ***an optimal solution for the instance*** $(\boldsymbol{U_i}, \boldsymbol{\mathcal{S}}, \boldsymbol{c})$.

- For any $1 \le j \le \ell$, define

$$\text{ap}'(O_j) \coloneqq \text{aprice}\left(O_j, (U - U_i) \cup \bigcup_{1 \le k < j} O_k\right).$$

Then it follows that, for any $1 \le j \le \ell$, we have

$$\text{aprice}(\widehat{S}_i, \mathcal{U} - U_i) \le \text{aprice}(O_j, \mathcal{U} - U_i) \le \text{ap}'(O_j) < \infty.$$

Guaranteed by our greedy choice.

By definition, the effective coverage of O_j in $\text{ap}'(O_j)$ is at most that in $\text{aprice}(O_j, \mathcal{U} - U_i)$.

Proof. (continue)

Now we prove the claim that $\text{aprice}(\widehat{S}_i, \mathcal{U} - U_i) \le \frac{OPT}{|U_i|}$.

Denote by $\mathcal{O} = \{O_1, O_2, \dots, O_\ell\}$ ***an optimal solution for the instance*** $(\boldsymbol{U_i}, \boldsymbol{\mathcal{S}}, \boldsymbol{c})$.

Then it follows that, for any $1 \le j \le \ell$, we have

$$\text{aprice}(\widehat{S}_i, \mathcal{U} - U_i) \le \text{aprice}(O_j, \mathcal{U} - U_i) \le \text{aprice}'(O_j) < \infty.$$

Then,

$$\text{aprice}(\widehat{S}_i, \mathcal{U} - U_i) \le \sum_{1 \le j \le \ell} \frac{\left|O_j - \bigcup_{1 \le k < j} O_k\right|}{|U_i|} \cdot \text{ap}'(O_j)$$

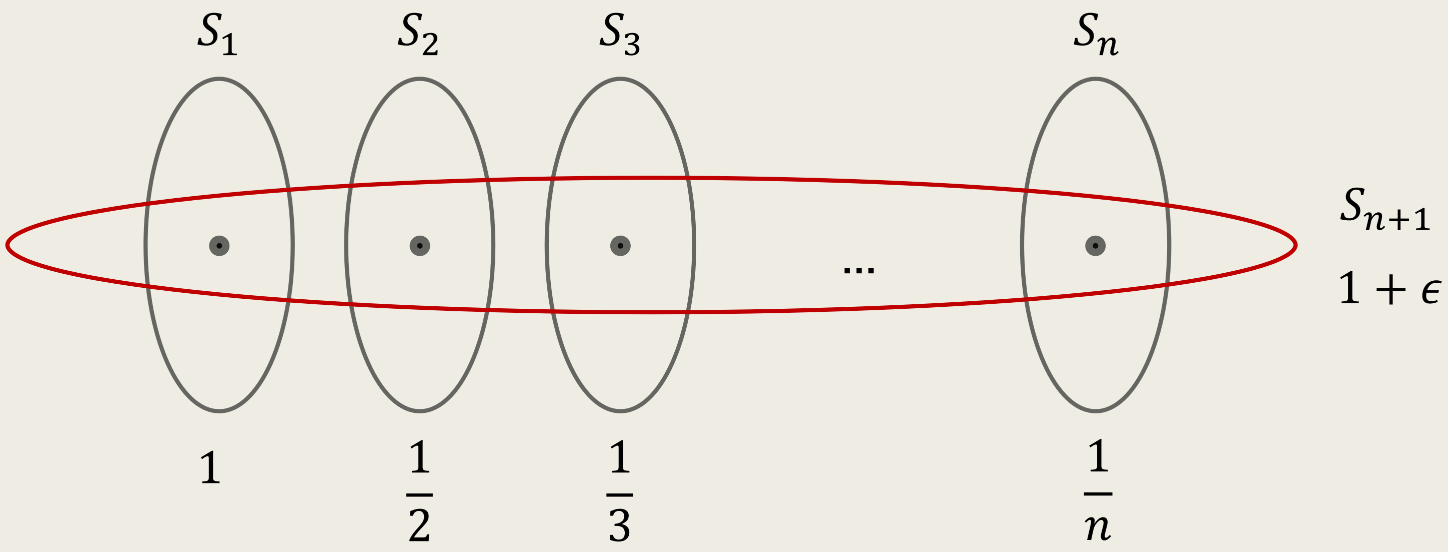
$$= \sum_{1 \le j \le \ell} \frac{1}{|U_i|} \cdot c(O_j) = \frac{c(\mathcal{O})}{|U_i|} \le \frac{OPT}{|U_i|}.$$

By the above inequality, and

$$\sum_{1 \le j \le \ell} \frac{\left|O_j - \bigcup_{1 \le k < j} O_k\right|}{|U_i|} = 1.$$

A Tight Example for the Greedy Algorithm

- The following example shows that,
  the approximation ratio of the greedy algorithm is indeed H_n.



The greedy algorithm will pick $S_1, S_2, \dots, S_n$, while the optimal solution is to pick S_{n+1}.

Randomized $O(\log n)$-Approximation for Set Cover via LP-rounding

Randomized Rounding for Set Cover

- We can use a simple & interesting randomized rounding technique to compute an $O(\log n)$-approximation for Set Cover.

  Consider the following natural ILP for set cover.

$$
\begin{aligned}
\min \quad & \sum_{A \in \mathcal{S}} w_A \cdot x_A & & (*) \\
\text{s.t.} \quad & \sum_{A \in \mathcal{S} : e \in A} x_A \geq 1, & & \forall\, e \in \mathcal{U}, \\
& x_A \in \{0, 1\}, & & \forall\, A \in \mathcal{S}.
\end{aligned}
$$

Randomized Rounding for Set Cover

1. Solve LP $(**)$ for an optimal fractional solution x^*.
2. Let $\mathcal{C} \leftarrow \emptyset$.

   Repeat the following process for $c \cdot \log n$ times.

   – For each $A \in \mathcal{S}$,

     include A into $\mathcal{C}$ with probability x_A^*.
3. Output $\mathcal{C}$.

We will set $c \coloneqq 1 + o(1)$.

$$\min \quad \sum_{A \in \mathcal{S}} w_A \cdot x_A \qquad (**)$$

$$\text{s.t.} \quad \sum_{A \in \mathcal{S} : e \in A} x_A \geq 1, \qquad \forall\, e \in \mathcal{U},$$

$$x_A \geq 0, \qquad \forall\, A \in \mathcal{S}.$$

The Feasibility

- Consider any $e \in \mathcal{U}$ and the sets $N(e) := \{ A \in \mathcal{S} : e \in A \}$ that contain e.
  - Consider *each* of the $c \cdot \log n$ iterations. We have

$$\Pr[\, e \text{ does not get covered} \,] = \prod_{A \in N(e)} (1 - x_A^*)$$

$$\leq \prod_{A \in N(e)} e^{-x_A^*} = e^{-\sum_{A \in N(e)} x_A^*}$$

$$\leq e^{-1}.$$

$1 + x \leq e^x$ holds for all $x \in \mathbb{R}$.

$\sum_{A \in N(e)} x_A^* \geq 1$ by the feasibility of x^* for LP $(**)$.

This process will cover $\mathcal{U}$ with high probability (w.h.p.).

The Feasibility

- Consider any $e \in \mathcal{U}$ and the sets $N(e) := \{ A \in \mathcal{S} : e \in A \}$ that contain e.
  - Consider each of the $c \cdot \log n$ iterations.

    We have $\Pr[\, e \text{ does not get covered} \,] \leq e^{-1}$.
  - Hence,

    $$\Pr[\, \mathcal{C} \text{ does not cover } e \,] \leq (e^{-1})^{c \cdot \log n} \leq \frac{1}{4n}$$

    for $c := 1 + o(1)$ such that $n^{-c} \leq 1/(4n)$.
  - Applying *union bound*, we get

    $$\Pr[\, \mathcal{C} \text{ does not cover } \mathcal{U} \,] \leq |\mathcal{U}| \cdot \frac{1}{4n} \leq \frac{1}{4}.$$

The Approximation Guarantee

- The expected cost incurred by each iteration is

$$E[\,\text{cost of subsets chosen in this iteration}\,] = \sum_{A \in \mathcal{S}} w_A \cdot x_A^* \;=\; OPT_f\,.$$

Hence, we have $E[\,w(\mathcal{C})\,] = c \cdot \log n \cdot OPT_f$.

- By Markov’s inequality, we get

$$\Pr\left[\,w(\mathcal{C}) \geq \; 4c \cdot \log n \cdot OPT_f\,\right] \leq \frac{1}{4}\,.$$

This cost is bounded with high probability (w.h.p.).

The Approximation Guarantee

- Combining the two w.h.p (with-high-probability) conclusions, it follows that

$$\Pr\Big[\ \mathcal{C} \text{ does not cover } \mathcal{U} \quad \text{or} \quad w(\mathcal{C}) \geq \ 4c \cdot \log n \cdot OPT_f\ \Big] \leq \frac{1}{2}\ .$$

- Repeat the entire process c' times for some constant $c' \in \mathbb{N}$ sufficiently large and output the best feasible solution.

  We get a $(4c \cdot \log n)$-approximation with probability at least $1 - 2^{-c'}$.

That’s all for Set Cover so far.

Let’s proceed to our next problem.