CSE515 Advanced Algorithms
Lecture 17
Greedy Algorithm for Set Cover

Antoine Vigneron
antoine@unist.ac.kr

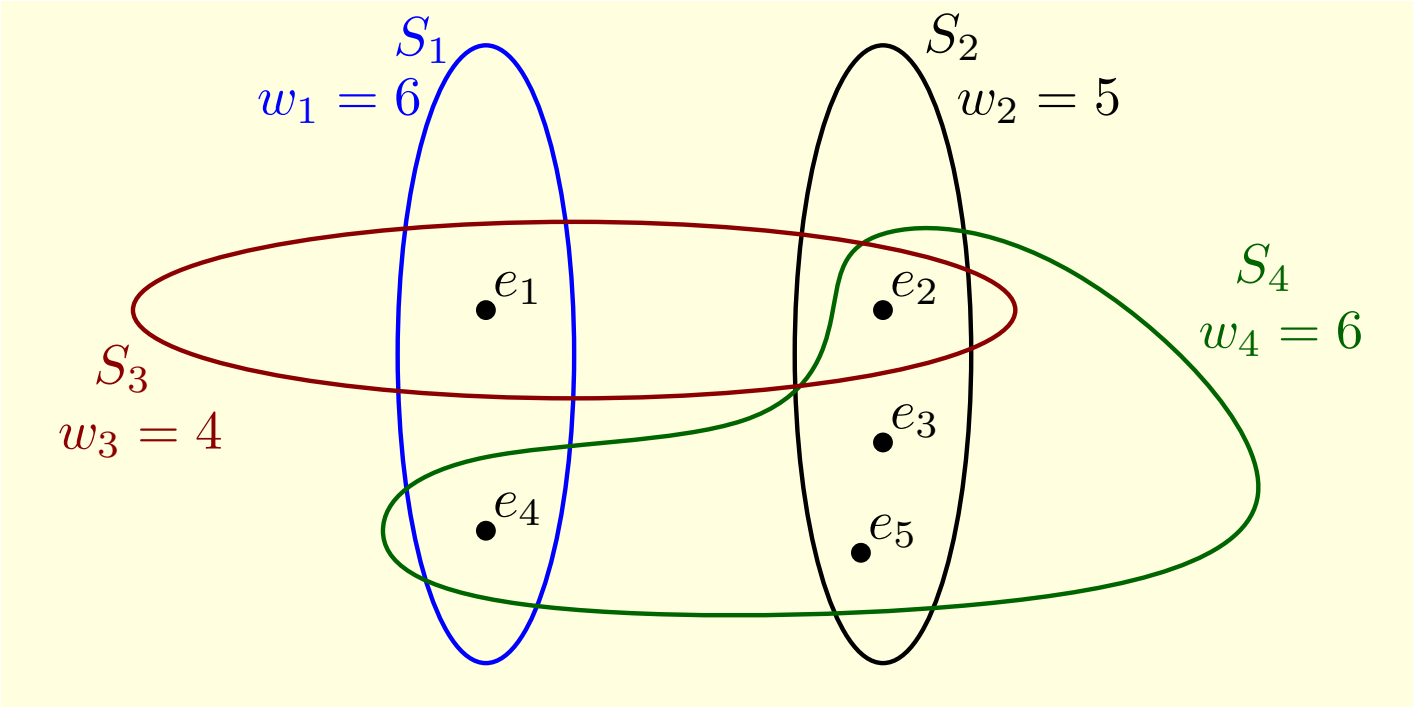
Ulsan National Institute of Science and Technology

April 27, 2021

Introduction

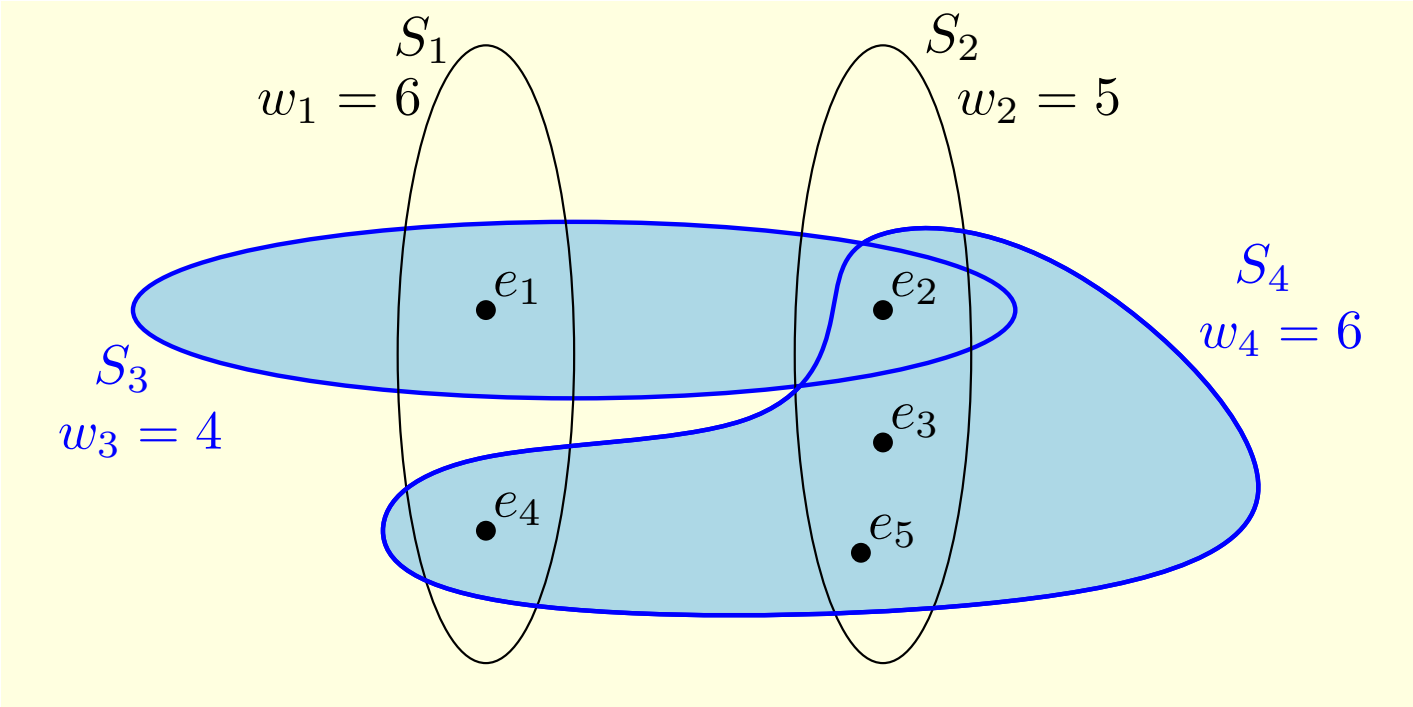
- Assignment 3 is due on Friday. at the beginning of the lecture.
- **Reference**:
  - Section 11.3 of the textbook Algorithm Design by Kleinberg and Tardos.
  - I changed some notations, i.e. s_i becomes e_i.

Problem Statement



- Problem: Find a cover of minimum weight.

Problem Statement



- Solution: S_3 and S_4, weight $w_3 + w_4 = 10$.

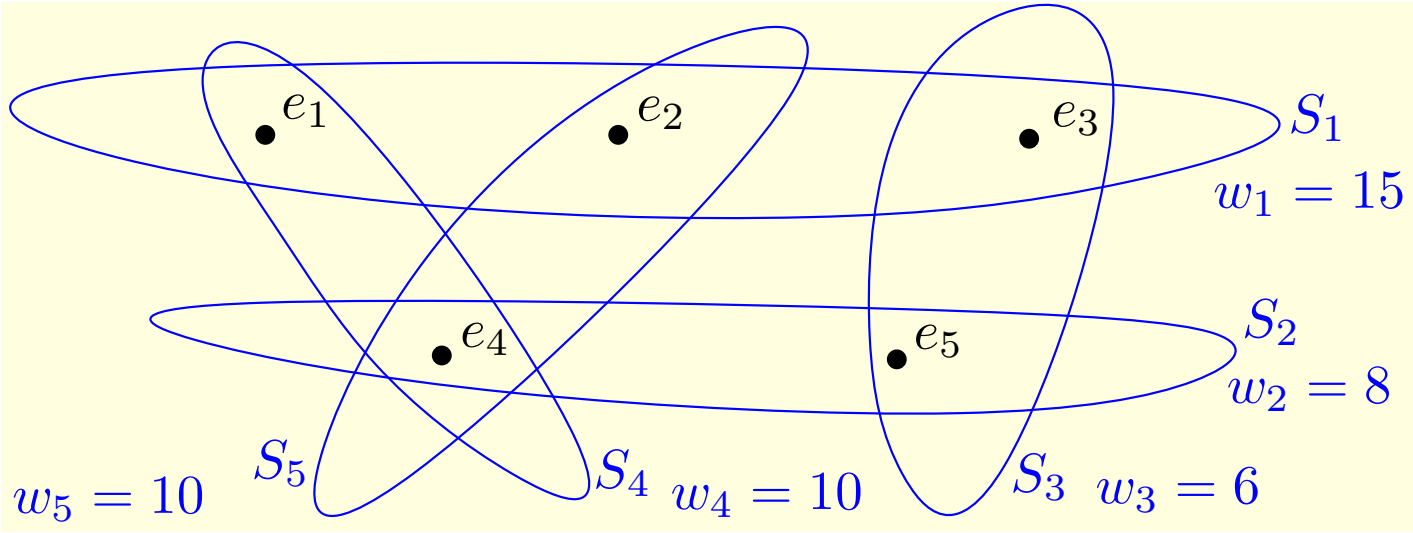
Problem Statement

Problem

We are given a set $U = \{e_1, \ldots, e_n\}$ *and subsets* $S_1, \ldots, S_m$ *of* U*, each subset* S_i *having a weight* $w_i > 0$*. A collection of subsets* $\mathcal{C} \subseteq \{S_1, \ldots, S_m\}$ *is a set cover if* $U = \bigcup_{S_i \in \mathcal{C}} S_i$*. The weight of a set cover is* $w(\mathcal{C}) = \sum_{S_i \in \mathcal{C}} w_i$*.*
The set cover problem is to find a set cover of minimum weight.

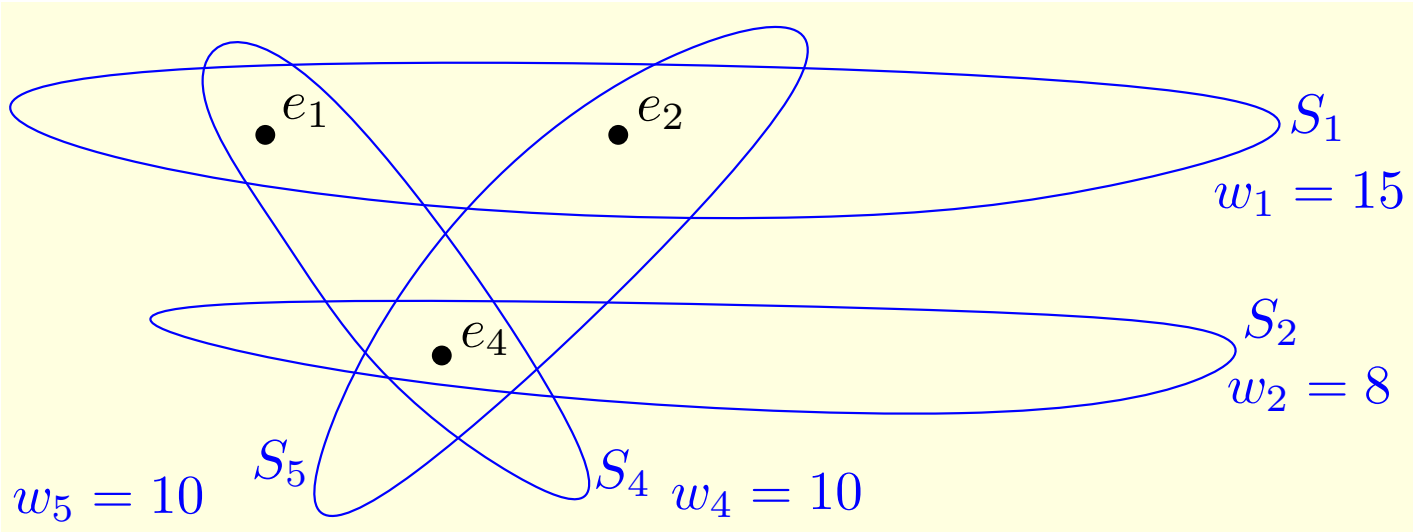
- Set cover is **NP**-hard. Why? (See lecture notes.)
- So we will try to find an approximation algorithm.

Greedy Algorithm



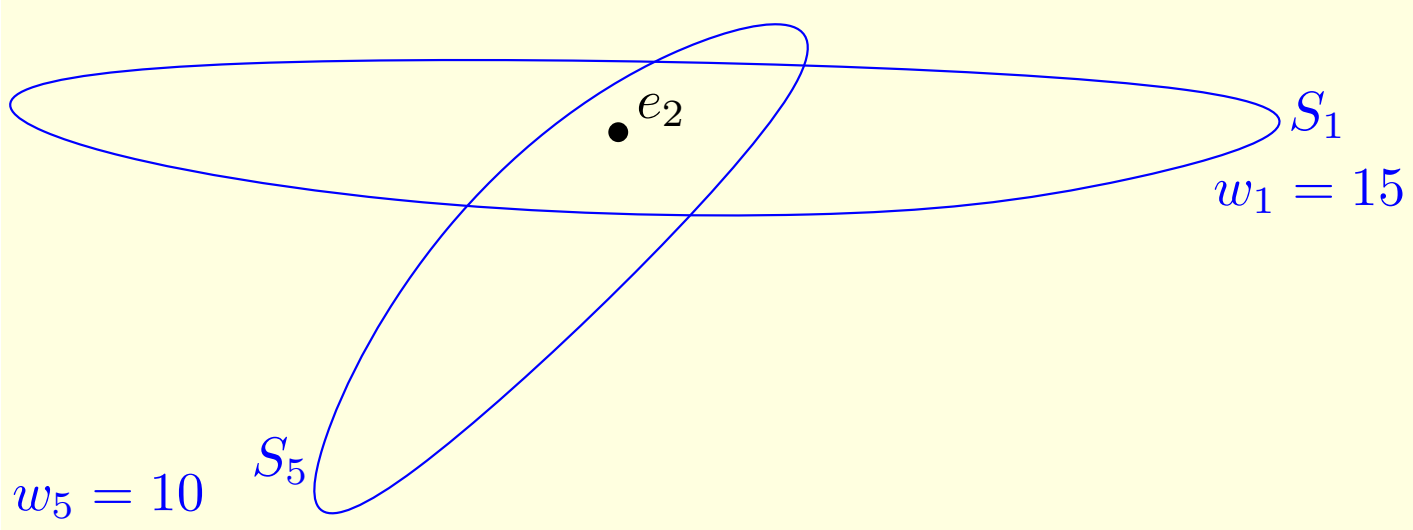
- S_1 has 3 elements and weight 15.
- So if I include S_1 in my set cover, I "pay" 5 for each of its elements e_1, e_2, e_3.
- If I include S_3, then I only pay 3 for e_3 and 3 for e_5.
- So I start by inserting S_3 in the cover.

Greedy Algorithm



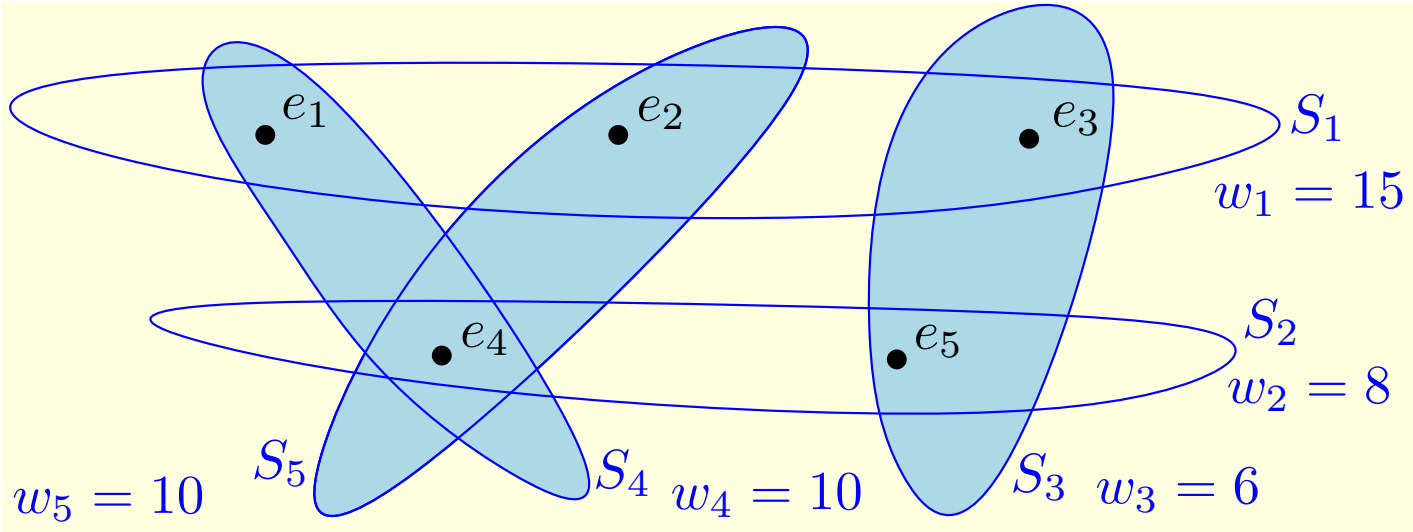
- I insert S_3 in the cover and remove its elements from U.
- Now the best choice is S_4 or S_5, with a cost of 5 per element covered. I choose S_4.

Greedy Algorithm



- I am left with only e_2, so I pick S_5.

Greedy Algorithm



- output: $\mathcal{C} = \{S_3, S_4, S_5\}$
- weight: $w(\mathcal{C}) = 26$
- The optimum is $\mathcal{C}^* = \{S_1, S_2\}$ with $w(\mathcal{C}^*) = 23$.

Pseudocode

Greedy algorithm for SET COVER

1: $R \leftarrow U$, $\mathcal{C} \leftarrow \emptyset$
2: **while** $R \neq \emptyset$ **do**
3: let S_i be the set that minimizes $w_i/|S_i \cap R|$
4: $R \leftarrow R \setminus S_i$
5: $\mathcal{C} \leftarrow \mathcal{C} \cup \{S_i\}$
6: **return** $\mathcal{C}$

- This algorithm is called *greedy* because it builds the solution step by step, trying to make as much progress as possible at each step.
- In particular, it does not look ahead: It does not try to see if the next steps will make much progress.
- Clearly this algorithm runs in polynomial time. We will now look at its approximation factor.

Analysis

- The nth *Harmonic number* is

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

  It is well known that $H_n = \ln(n) + O(1)$.

- We will prove that the greedy algorithm is an H_n-approximation algorithm for set cover.
- More precisely:

Theorem

Let $d^* = \max_i |S_i|$ be the maximum size of the subsets. The greedy algorithm is an H_{d^*}-approximation algorithm for set cover.

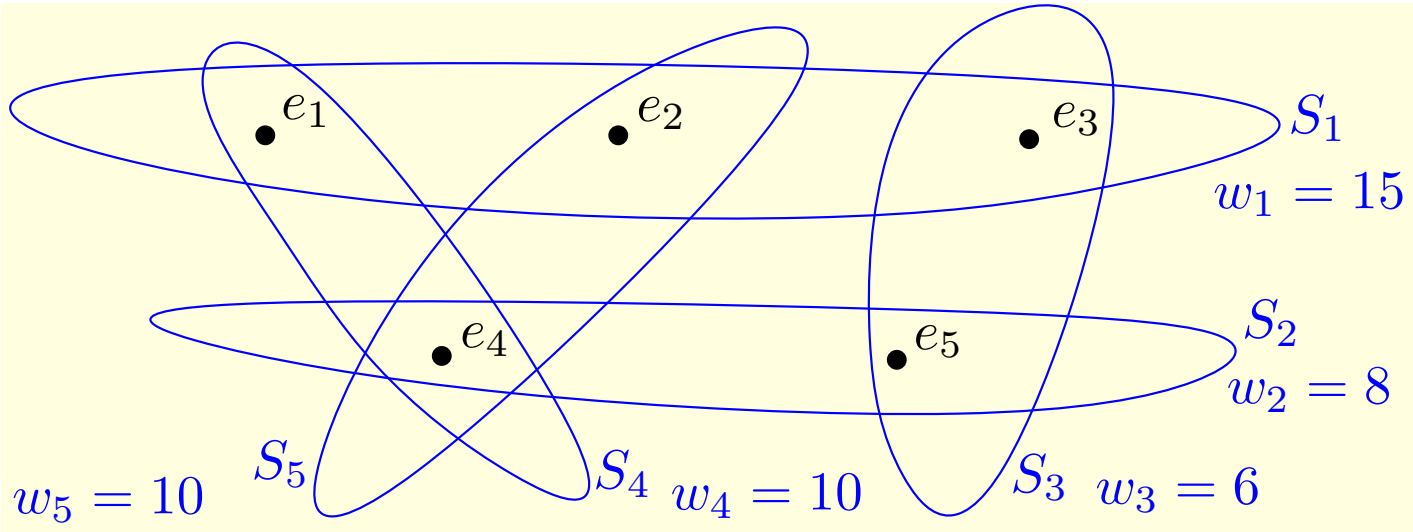
Analysis

- As we mentioned in the algorithm description, when we insert S_i in the solution, we may consider that we pay a price $w_i/|R \cap S_i|$ for each element of $R \cap S_i$.
- So for each $e \in R \cap S_i$, we set

$$c_e = \frac{w_i}{|R \cap S_i|}$$

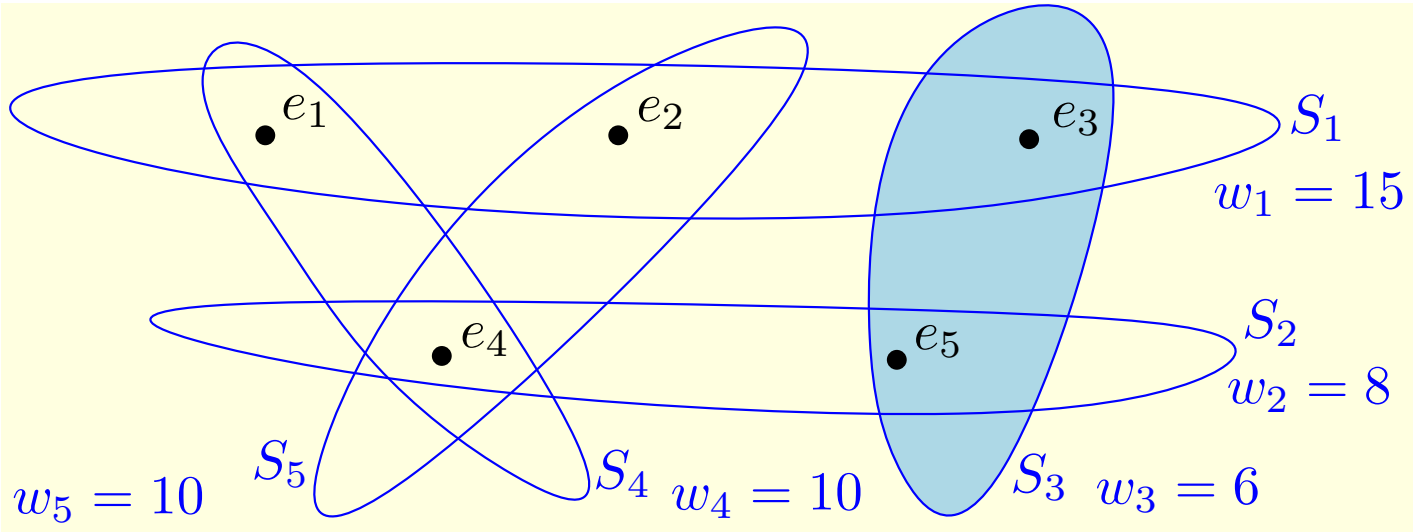
  at the time e is covered (for the first time) by a set S_i.
- An example is given in next slides.

Constructing y



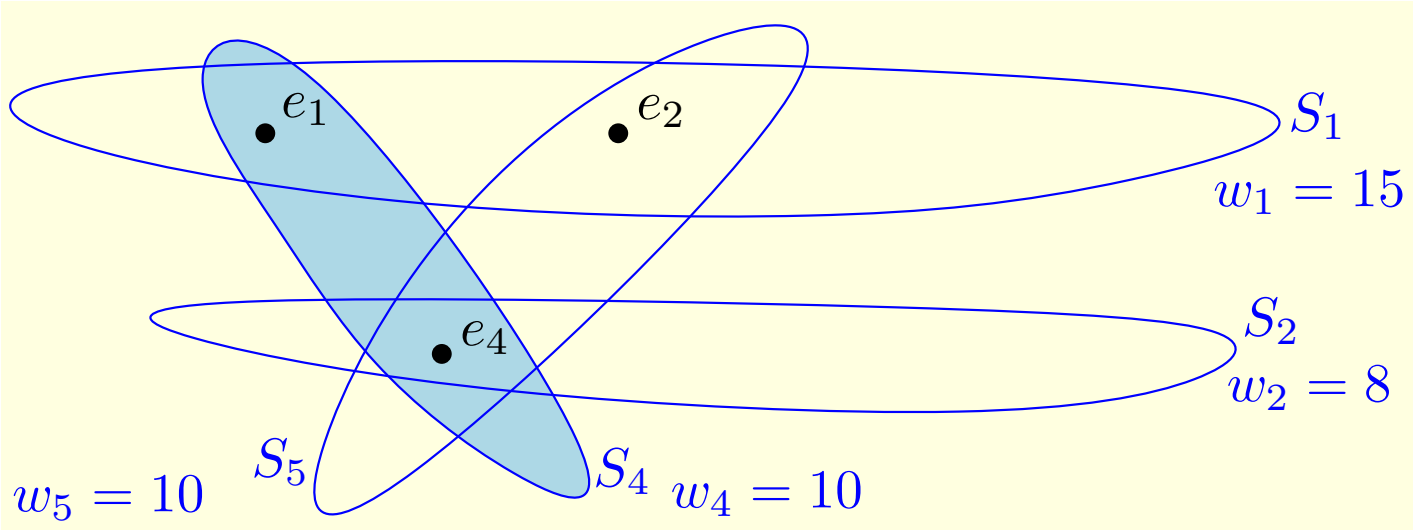
c_{e_1}	0
c_{e_2}	0
c_{e_3}	0
c_{e_4}	0
c_{e_5}	0

Constructing y



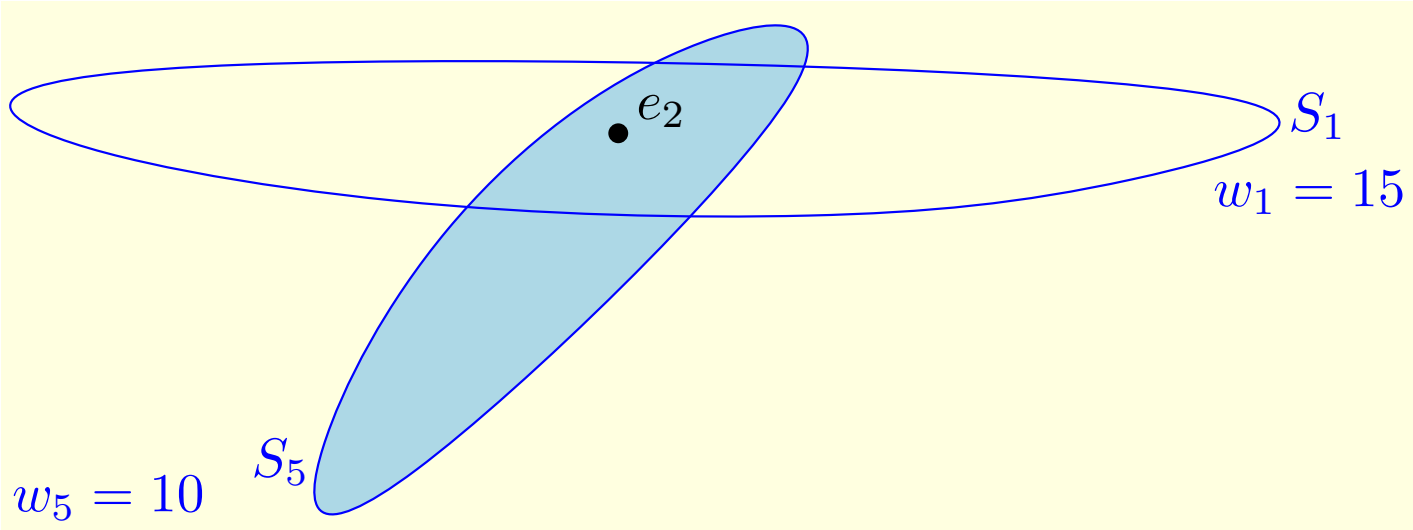
c_{e_1}	0	0
c_{e_2}	0	0
c_{e_3}	0	3
c_{e_4}	0	0
c_{e_5}	0	3

Constructing y



c_{e_1}	0	0	5
c_{e_2}	0	0	0
c_{e_3}	0	3	3
c_{e_4}	0	0	5
c_{e_5}	0	3	3

Constructing y



c_{e_1}	0	0	5	5
c_{e_2}	0	0	0	10
c_{e_3}	0	3	3	3
c_{e_4}	0	0	5	5
c_{e_5}	0	3	3	3

Analysis

- Each time a set S_i is inserted in the solution, the total costs of the newly covered elements is w_i. It follows that:

Lemma

The total weight of the solution $\mathcal{C}$ returned by the greedy algorithm is $w(\mathcal{C}) = \sum_{e \in U} c_e$

- The lemma below shows how to handle each set S_k separately. (Proof done in class, see textbook.)

Lemma

Let $k \in \{1, \ldots, m\}$ and $d = |S_k|$. Then $\sum_{e \in S_k} c_e \leqslant H_d \cdot w_k$.

- We can now complete the proof of the theorem on Slide 12. (Done in class, see textbook.)

Analysis

- Is our analysis tight: Could it be that the greedy algorithm has an approximation factor less than $o(\log n)$ in the worst case?
- An example given in the textbook shows that it is impossible, i.e. the greedy algorithm is a factor about $\log n$ from optimal on this input.

Concluding Remarks

- We gave a greedy algorithm for SET COVER.
- Its approximation factor is logarithmic.
- If the sets have bounded size, i.e. $\max_i |S_i| = O(1)$, then it is a constant factor.
- Some hardness results in complexity theory suggest that we cannot hope to obtain a sub-logarithmic approximation factor in the worst case.
- Greedy algorithms can be applied to many problems. They are often used as heuristics, i.e. even if we cannot prove a good approximation factor.
- Usually greedy algorithms do not provide an optimal (exact) solution, but there are some exception, for instance a minimum spanning tree (MST) can be obtained by a greedy algorithm.