

Set Cover

Set System: $\Sigma = (X, S)$ where

$X = \{x_1, \dots, x_n\}$ domain of things to be covered.

$S = \{s_1, \dots, s_m\}$ collection of sets $s_i \subseteq X$

that can be used to build the cover

Set Cover Problem: Given a set system (X, S) ,

find smallest number of sets that cover all of X

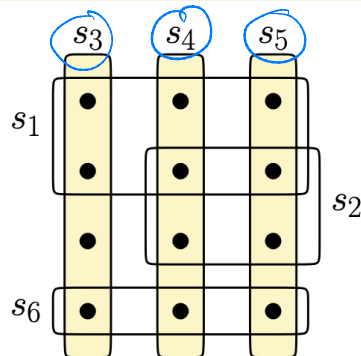
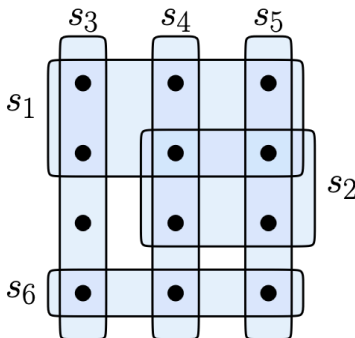
$C \subseteq \{1, \dots, m\}$ of min size s.t.

$$\bigcup_{i \in C} s_i = X$$

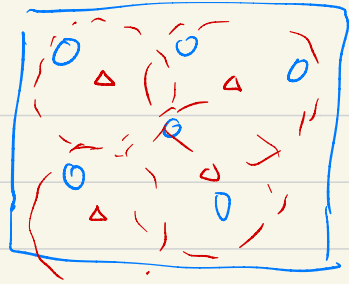
$\{3, 4, 5\}$ is a cover

$$n=12$$

$$m=6$$



Application: Wireless Coverage.



Workforce building



Greedy Algorithm: pick the set that covers the most uncovered items.

greedy-set-cover (X, S)

$U \leftarrow X$

U : all element yet to cover

$C \leftarrow \emptyset$

C : current selection.

while ($U \neq \emptyset$) $\leftarrow O(n)$

$O(m) \cdot O(n)$

$O(n^2 \cdot m)$

$i \leftarrow$ set s_i covers most elements of U .

$C \leftarrow C \cup \{i\}$

select i w/ min $u_i \leftarrow O(m)$

$U \leftarrow U \setminus s_i$

$\rightarrow$ update u_i $O(m)$ per each element newly covered

return C .

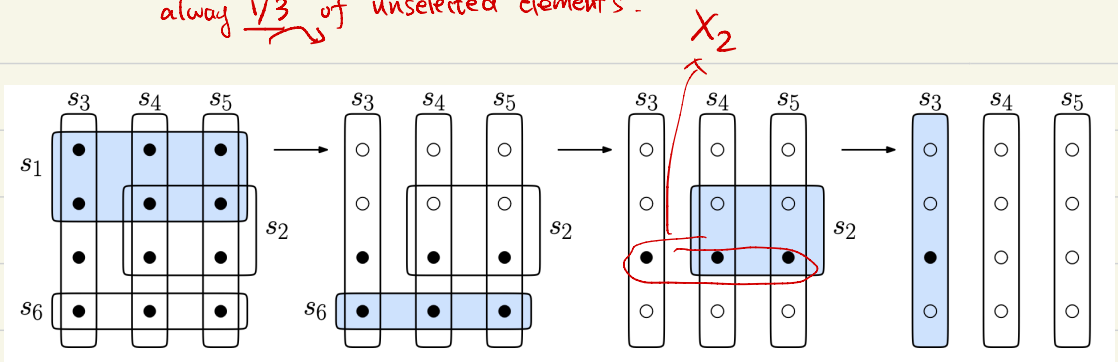
$O(n)$ such elements

Running time: $O(nm)$ why?

maintain u_i : # elements in U set covered by s_i

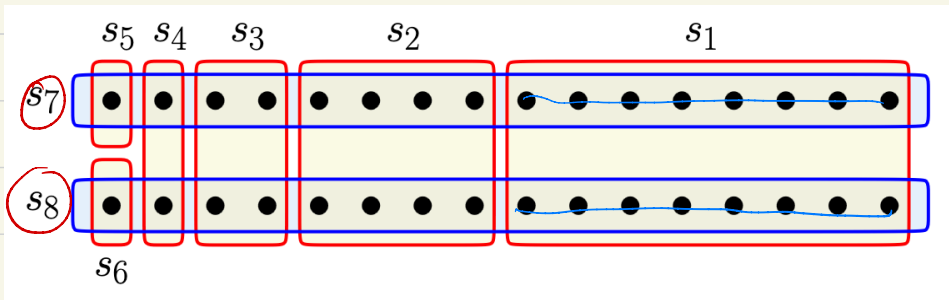
$$w = 3.$$

always $\frac{1}{3}$ of unselected elements.



Greedy: $\{1, 6, 2, 3\}$ Optimal: $\{3, 4, 5\}$

Optimality? No.



Optimal: $\{7, 8\}$.

Greedy: s_1, s_7, s_8 all cover 16 points

$\Rightarrow$ pick s_1 .

$s_2, s_7, s_8 \dots 8$ points

$\Rightarrow$ pick s_2

$\vdots$

$\{1, 2, 3, 4, 5, 6\}$

2 lines of 2^n points

Opt: 2. Greedy: $n+2$

Gap between Greedy and Optimal $\sim O(\log n)$

Approximation:

Theorem: Given any set system (X, S) , let

O = optimal set cover, G = greedy set cover,

$$|G| \leq |O| \cdot \ln n + 1$$

removable, longer proof!

Proof. $g = |G|$, $o = |O|$

Let $X_i = \{\text{items that are still uncovered after } i^{\text{th}} \text{ iteration}\}$

$$n_i = |X_i|, \quad (n_0 = n) \quad X_0 = X, X_1, X_2, \dots, X_g = \emptyset$$

At the start of i^{th} round:

need to cover n_{i-1} items in current & later rounds

$\exists$ a cover of X_{i-1} , that has size $\leq o$

$\Rightarrow \exists$ a set covers n_{i-1}/o items of X_{i-1}

pigeon hole principle: o sets $\rightarrow$ cover n_{i-1} items
 $\Rightarrow \exists$ one set $\rightarrow$ cover $\frac{n_{i-1}}{o}$ items

$\Rightarrow$ the set S_i picked by greedy must cover

at least n_{i-1}/o items.

$$\Rightarrow n_i \leq n_{i-1} - n_{i-1}/o = n_{i-1} \left(1 - \frac{1}{o}\right)$$

$$\Rightarrow n_i \leq n_{i-1} \left(1 - \frac{1}{o}\right) \leq n_{i-2} \left(1 - \frac{1}{o}\right)^2 \leq \dots \leq \frac{n_0}{n} \left(1 - \frac{1}{o}\right)^i$$

At iteration $g-1$, still some uncovered items.

$$n_{g-1} \geq 1$$

$$\Rightarrow 1 \leq n_{g-1} \leq n \cdot \left(1 - \frac{1}{\sigma}\right)^{g-1}$$

Lemma: $\left(1 - \frac{1}{\sigma}\right) \leq e^{-1/\sigma}$

$$1 \leq n \left(1 - \frac{1}{\sigma}\right)^{g-1} \leq n \cdot e^{-1/\sigma \cdot (g-1)}$$
$$= n / e^{(g-1)/\sigma}$$

$$\Rightarrow n \geq e^{(g-1)/\sigma}$$

$$\Rightarrow \ln n \geq (g-1)/\sigma$$

$$\Rightarrow g \leq \sigma \cdot \ln n + 1$$

Optimal Solution?

NP-hard.

later lecture.

$(1+\epsilon)$ - approx

2 - approx

c - approx

$(1-e) \cdot \ln n$ - approx

NP-hard. $\rightarrow$ Greedy is the best!

Stone Merging

Problem: you have n piles of stones.
each has $a_1, a_2, \dots, a_n$ stones.

you can choose 2 piles, merge them together.
cost is total number of stones of the two piles

Goal: minimal cost to merge all of them.

Example:

Start.

1 2 3 4 5

merge 2 and 4 → 6
1 3 5 6

merge 3 and 5 → 8
1 6 8

merge 1 and 6 → 7

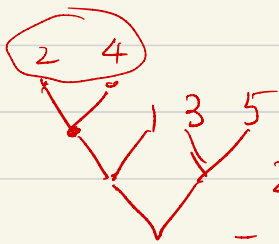
merge 7 and 8 → 15

cost: $2+4=6$

cost: $3+5=8$

cost: $1+6=7$

cost: $7+8=15$



$2 \times 3 + 4 \times 3 + 1 \times 2 + 3 \times 2 + 6 \times 2$ total cost $6+8+7+15$
 $= 36$ $= 36$

Background: Optimal encoding (Huffman code problem)

$a, b, c, \dots, z$ alphabet, character $\rightarrow$ 0/1 string
 $a \rightarrow 00$ $b \rightarrow 0101$ $c \rightarrow 111$, ...

what is the most efficient encoding?

Greedy:

Idea: always merge the smallest first.

Algorithm: easy,

Time complexity: $O(n \log n)$ using binary heap as priority queue.

Optimality?

Proof: $\mathcal{O}$ optimal solution. $\mathcal{G}$: greedy.

$\mathcal{O}$: and $\mathcal{G}$, agrees on first k steps.

At $(k+1)$ step, both $\mathcal{O}$ & $\mathcal{G}$ face the same piles:

$(n-k)$ piles: $b_1, \dots, b_{n-k}$

Plan: adjust $\mathcal{O}$, s.t. smallest 2 piles in $\{b_1, \dots, b_{n-k}\}$ merged the first, without losing optimality.

$b_1 \quad b_2 \quad \dots \quad b_{n-k}$

Merging plan: binary tree, with b_i 's as leaf node.

$$\text{total cost} : \sum_i b_i \cdot \text{depth}(i)$$

First merge : nodes with highest depth, they are siblings.

Let b_x, b_y to be the piles with small sizes.

Adjust b_x, b_y to somewhere

If b_x & b_y already in ^(the) somewhere, good!

If not, b_z, b_w are in that place.

Adjustment: swap $b_x \leftrightarrow b_z$, $b_y \leftrightarrow b_w$

$$b_x \leq b_z, \quad b_y \leq b_w$$

$$\text{depth}(b_x) \leq \text{depth}(b_z), \quad \text{depth}(b_y) \leq \text{depth}(b_w)$$

Before: cost: $b_x \cdot \text{depth}(b_x) + b_z \cdot \text{depth}(b_z) + \dots$

After: $b_x \cdot \text{depth}(b_z) + b_z \cdot \text{depth}(b_x) + \dots$