Today:

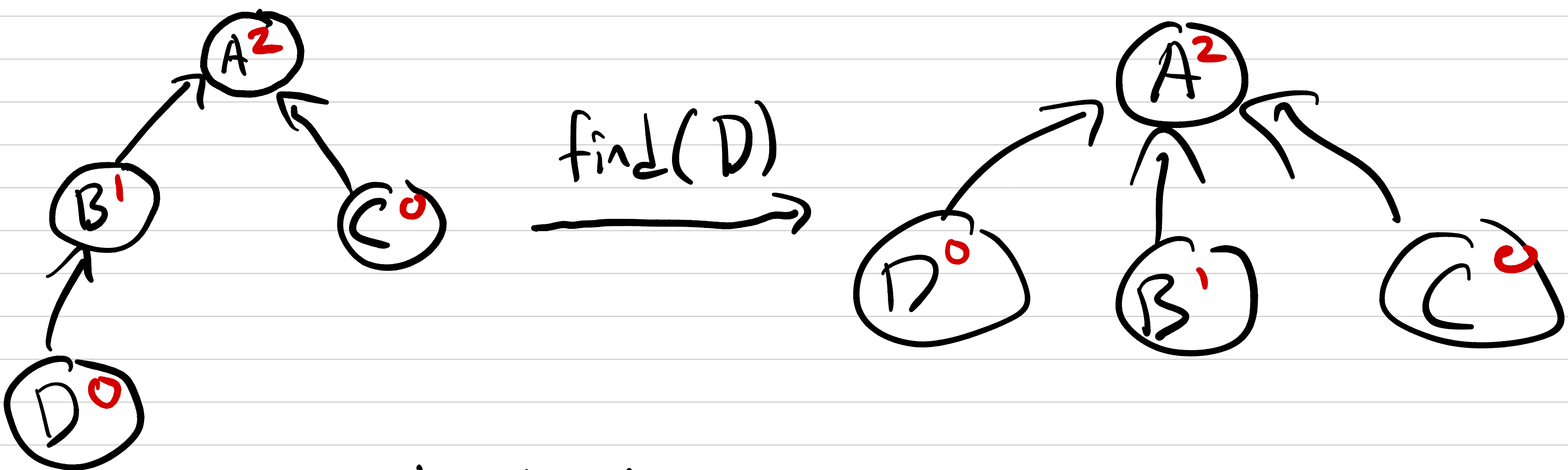
1. Finish union find

2. Set cover

3. Dynamic Programming?

Recap: Union find data structure (with path compression)

- makeset (x) • find (x) • merge (x,y)



Last time: makeset $O(1)$ time
find / union : $O(\log(n))$ (without path compression)

Today: A sequence of n makesets and m union/find ops takes $O((n+m)\log^*(n))$ time
(avg operation takes $\log^* n$ time) (w/ path compression)

Key idea: Group elements x by $\text{rank}(x)$

Intervals: "-1" "0" "1" "2" "3" "4"

$[0]$ $[1]$ $(1,2]$ $(2,4]$ $(4,16]$ $(16,65536]$

For $i \geq 1$, interval i is $(2\uparrow(i-1), 2\uparrow i]$, $2\uparrow i = \left.2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}\right\}i$

Def: Element x is interval i at some time if:

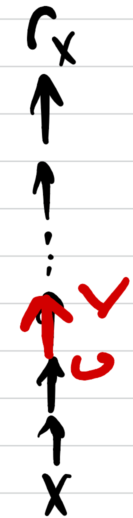
- x is not a root
- x's $\text{rank}(x)$ is in interval i

Two facts: 1. Largest nonempty interval is $i = \log^*(n) - 1$

2. # of elements in interval i is $\leq \frac{n}{2\uparrow i}$

Consider find(x). Cost is # of edges in

Consider $\underset{u}{\overset{v}{\uparrow}}$ in this path. 3 types of edges:



Type 1: v is a root, $O(1)$ cost per find

Type 2: v is not a root, u's interval $\neq$ v's interval

$O(\log^*(n))$ type 2 edges per find

Type 3: v is not a root, u's interval = v's interval

Can be a lot! "u's type 3 edge"

Cost of type 3 edges

$$\sum_{\text{element } u} (\#\text{ of times } u \text{ is type 3 edge})$$

Suppose u is in interval i

rank(u) $\quad v \quad v'$

$$(2\uparrow(i-1),\ 2\uparrow(i-1)+1,\ \ldots\ \ldots,\ 2\uparrow i]$$

u is type 3 edge $\leq 2\uparrow i$ times

cost $\leq \displaystyle\sum_{\text{elements } u} \underbrace{2\uparrow i}_{u\text{'s interval}} = \sum_{\substack{\text{intervals} \\ i=-1}}^{\log^*(n)-1} 2\uparrow i \cdot (\#\text{ elements in interval})$

$$\leq \sum_i 2\uparrow i \,\frac{n}{2\uparrow i} = \sum_i n = O(n \log^* n)$$

$\square$

In summary

For each find, type 1 & 2 edges cost $O(\log^*(n))$

Over all finds, type 3 edges cost $O(n \log^*(n))$

Total cost $= O(m \cdot \log^*(n) + n \log^*(n))$

(average $\log^*(n)$ per operation)

Recall: can do even better! average $\alpha(n+m, n)$ / operation

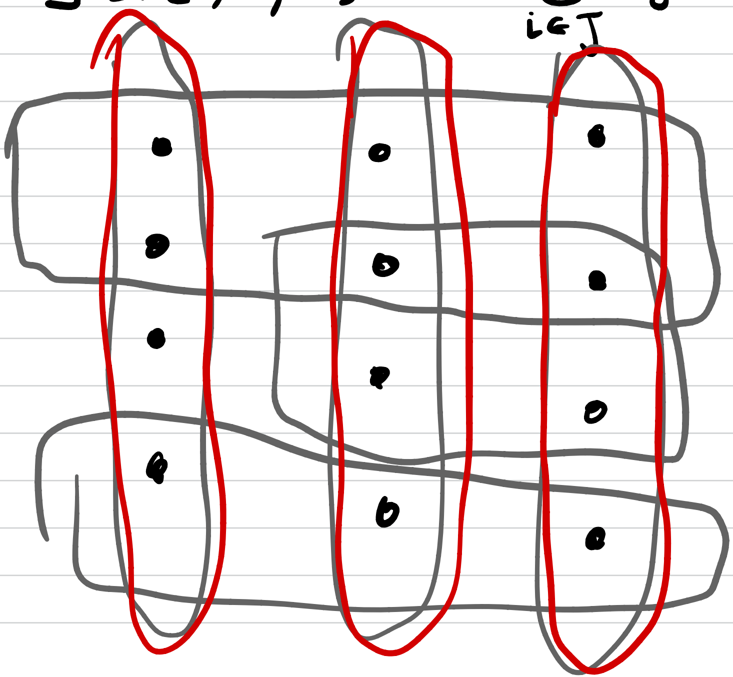
inverse Ackerman

Set Cover

Input:
- Universe of n elements $\mathcal{U} = \{1, \ldots, n\}$
- Subsets $S_1, \ldots, S_m \subseteq \mathcal{U}$ s.t. $S_1 \cup \cdots \cup S_m = \mathcal{U}$

Output: Collection of these subsets covering $\mathcal{U}$ of minimal size k
i.e. $J \subseteq \{1, \ldots, m\}$ s.t. $\bigcup_{i \in J} S_i = \mathcal{U}$

Example:



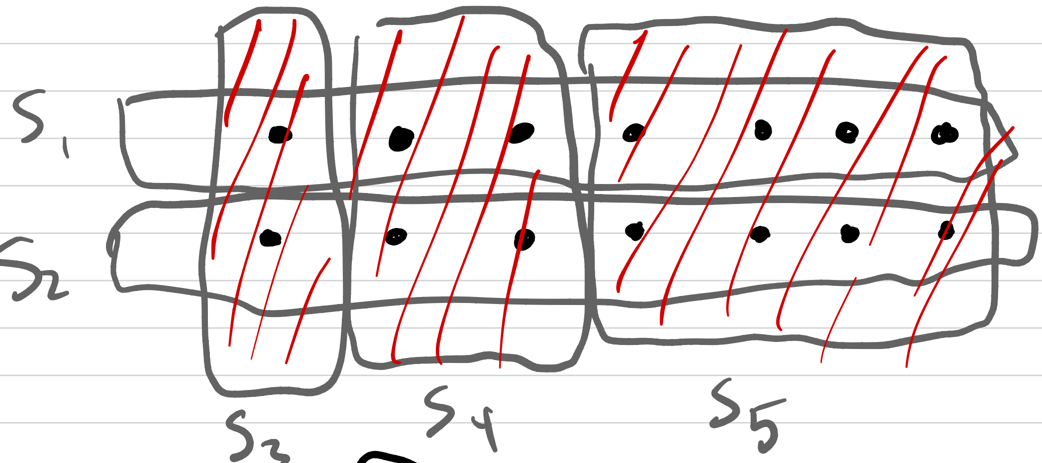
Minimal set cover size

$k = 3$

Greedy algorithm for set cover

Repeat until all elements of $\mathcal{U}$ are covered:

Pick the next set S with largest # of uncovered elements

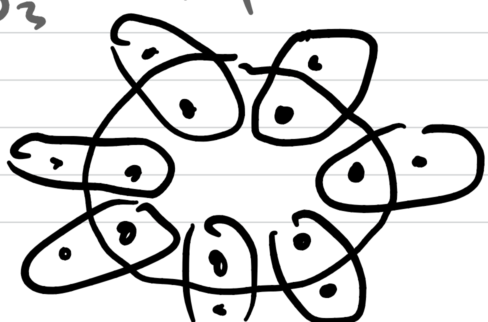


Greedy picks: S_5, S_4, S_3

Optimal: S_1, S_2

Opt size $k=2$

Greedy fails! :(

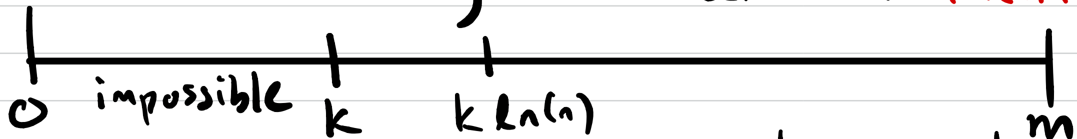


Greedy picks all 8 sets

Optimal just the petals, $k=7$

Theorem: "Greedy solution is not too bad"

If optimal solution uses k sets, then Greedy uses at most $k\ln(n)$ sets

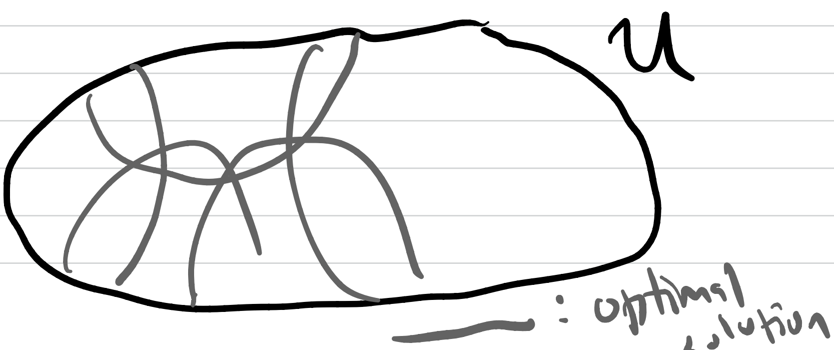


Pf: Let n_t be # of elements not covered after t steps of Greedy. (E.g. $n_0 = n$)

Goal: show for $t = k\ln(n)$, $n_t < 1$

$\Rightarrow n_t = 0$, Greedy covers $\mathcal{U}$ w/ t sets.

Claim 1: $n_1 \le n_0 - n_0/k$



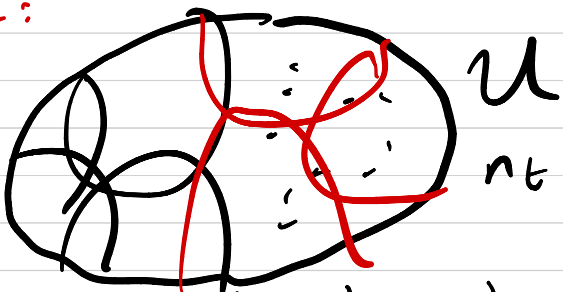
Optimal solution uses k sets

Therefore, $\exists$ set covering $\geq n/k$ elements

Greedy picks largest set, which is of size $\geq n/k$ $\blacksquare$

Claim 2: $n_{t+1} \le n_t - \frac{n_t}{k} = n_t\left(1 - \frac{1}{k}\right)$

Pf:



—— : sets chosen by Greedy

—— : sets chosen by optimal

Optimal solution covers these n_t points

$\Rightarrow \exists$ a set that covers $\geq n_t/k$ of these points

Greedy picks largest set, covers $\geq \frac{n_t}{k}$ remaining points $\square$

$$n_t \le n_{t-1}\left(1 - \frac{1}{k}\right) \le n_{t-2}\left(1-\frac{1}{k}\right)^2 \le \dots \le n_0\left(1 - \frac{1}{k}\right)^t$$
$$= n\left(1 - \frac{1}{k}\right)^t$$

Question: for $t = k\ln(n)$, can we show

$$n\left(1 - \frac{1}{k}\right)^t < 1 \,?$$

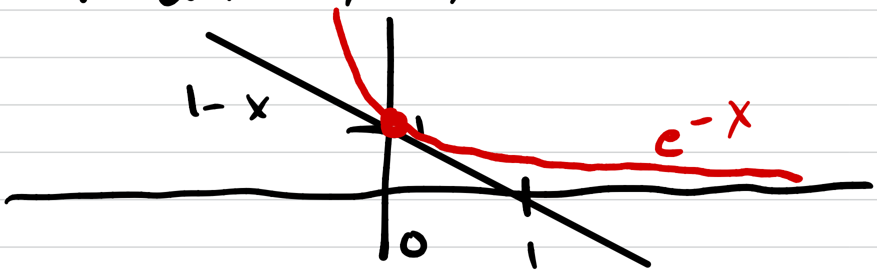
We have shown: $n_t \le n\left(1-\frac{1}{k}\right)^t$

[Intuition: $\left(1-\frac{1}{k}\right)^k \approx 1/e$

$\left(1-\frac{1}{k}\right)^t = \left(\left(1-\frac{1}{k}\right)^k\right)^{t/k} \approx \left(1/e\right)^{t/k}$]

Fact: For all $x \neq 0$, $1-x < e^{-x}$

Pf:



$\square$

$$n_t \le n\left(1-\frac{1}{k}\right)^t \underset{1-x<e^{-x}}{<} n\left(e^{-1/k}\right)^t = n e^{-t/k}$$

When $t = k\ln(n)$, $n_t < n e^{-k\ln(n)/k} = n e^{-\ln(n)}$

$$= \frac{n}{n} = 1 \qquad \square$$

Greedy does not solve set cover

Greedy outputs $k \ln(n)$ sets, where $k = OPT$

"Greedy achieves approximation $\ln(n)$"

We don't know any efficient alg for set cover.

But we do "know" the best efficient approximation alg

It achieves approximation $\ln(n)$

It is Greedy!

Thm: If $\exists$ an efficient alg for set cover with approximation $0.999 \ln(n)$, then

- then would be efficient alg for set cover
- " " " " " " factoring
- " " " " " " 3sat
- $P = NP$