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 \( \Theta(1) \) time
find/union: \( \Theta(\log(n)) \) (without path compression)

Today: A sequence of \( n \) makesets and \( m \) union/find ops takes \( \Theta((n + m) \log^*(n)) \) time
(avg operation takes \( \log^* \) 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, \infty] \]

For \( i \geq 1 \), interval \( i \) is \( (2^{2i-1}, 2^{2i}] \), \( 2^{2i} = \frac{z^i}{z} \)

Def: Element \( x \) is in interval \( i \) at some time if:

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

Two facts:
1. Largest nonempty interval is \( 0 = \log^* n - 1 \)
2. \# of elements in interval \( i \)

\( is \leq \frac{n}{2^{2i}} \)
Consider \( \text{find}(x) \). Cost is \# of edges in

Consider \( \uparrow \) 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

\[ \sum_{\text{element } u} \left( \text{# of times } u \text{ is type 3 edge} \right) \]

Suppose \( u \) is in interval \( i \)

\( (2^{\log_{2}(u)-1}, 2^{\log_{2}(u)-1}+1, \ldots, 2^{\log_{2}(u)}] \)

\( u \) is type 3 edge \( \leq 2^{\log_{2}(u)} \) times

\[ \text{cost} \leq \sum_{\text{elements } u} 2^{\log_{2}(u)} = \sum_{\text{intervals } i} 2^{\log_{2}(u)} \cdot (\text{# elements in interval}) \]

\( u \)'s interval

\[ \leq \sum_{i} 2^{\log_{2}(u)} \frac{n}{2^{\log_{2}(u)}} = \sum_{i} n = O(n \log n) \]

\( \Box \)
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 $\Theta(n^{-n}/\text{opti})$

inverse Ackerman
Set Cover

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

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

Example:
Minimal set cover size $k = 3$
Greedy algorithm for set cover

Repeat until all elements of \( U \) 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! \( n \)

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 the # of elements not covered after $t$ steps of Greedy. (E.g. $n_0 = n$)

Goal: show for $t = k \ln(n)$, $n_t < 1$ implies $n_t = 0$, Greedy covers $U$ w/ $t$ sets.

Claim 1: $n_t \leq n_0 - n_0/k$

Optimal solution uses $k$ sets

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

Greedy picks largest set, which is of size $\geq n/k$
Claim 2: \( n_{t+1} \leq n_t - \frac{n_t}{k} = n_t \left(1 - \frac{1}{k}\right) \)

Optimal solution covers these \( n_t \) points.

\( \exists \exists a \) set that covers \( \geq \frac{n_t}{k} \) of these points.

Greedy picks largest set, covers \( > \frac{n_t}{k} \) points.

\( n_t \leq n_{t-1} \left(1 - \frac{1}{k}\right) \leq n_{t-2} \left(1 - \frac{1}{k}\right)^2 \leq \cdots \leq n_0 \left(1 - \frac{1}{k}\right)^t = n \left(1 - \frac{1}{k}\right)^t \)

Question: for \( t = k \ln(n) \), can we show \( n \left(1 - \frac{1}{k}\right)^t < 1 \)?
We have shown:  \( n_t \leq n \left( 1 - \frac{1}{k} \right)^t \)

\[
\begin{align*}
\text{Intuition:} & \quad (1 - \frac{1}{k})^k \approx \frac{1}{e} \\
& \quad (1 - \frac{1}{k})^t = \left( (1 - \frac{1}{k})^k \right)^{t/k} \approx \left( \frac{1}{e} \right)^{t/k}
\end{align*}
\]

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

Pf:

When \( t = k \ln(n) \), \( n_t < n e^{-k \ln(n) / k} = n e^{-\ln(n)} = \frac{n}{n} = 1 \)
Greedy does not solve set cover
Greedy outputs $k \ln(n)$ sets, where $k = O(P_T)$
"Greedy achieves approximation $\ln(n)$"

We don't know any efficient alg for set cover.
But we do "know" the best efficient approximation
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
- $P = NP$