OH TODAY: 1:15 pm - 2:15 pm.

Union-Find:

Motivation: Kruskal’s algorithm. When we process an edge e = (u, v), we need to check if u and v are in different connected components or not.

We need a data structure that will maintain disjoint sets where each set represents a CC in our (partial) solution.

Operations:

makeSet(x): create a singleton set containing x

find(x): returns the name of the set to
which $x$ belongs to

Union $(x, y)$ : merge the sets containing $x$ and $y$.

Kruskal:

- for each $u \in V$ do
  - makeSet $(u)$
  \[ O(n) \]
- \[ X \leftarrow \{y\} \] // solution \[ \rightarrow O(1) \]
- Sort edges in order of wt. \[ \rightarrow O(m \log n) \]
- for each $e = (u, v)$ do
  - if $\text{find } (u) \neq \text{find } (v)$ then
    - add $e$ to $X$
    - $\text{union } (u, v)$
  \[ O(m \log n) \]
- return $X$
Disjoint sets maintained as a directed tree.
Each node in the directed tree has
- parent pointer
- rank, which for now denotes the
  height of the tree rooted at that node.

root of the tree is the representative/handle
of the set.

\[
\text{make-set}(x) \\
\pi(x) \leftarrow x \\
\text{rank}(x) \leftarrow 0
\]

\[
\text{find}(x) \\
\begin{cases}
\text{while } x \neq \pi(x) \text{ do} \\
x \leftarrow \pi(x) \\
\text{return } x
\end{cases}
\]

To make find efficient, we need trees
to be shallow.

Union \((x, y)\)

\[ r_x \leftarrow \text{find}(x) \]
\[ r_y \leftarrow \text{find}(y) \]

if \(r_x = r_y\) then return

else if \(\text{rank}(r_x) > \text{rank}(r_y)\) then

\[ \pi(r_y) = r_x \]

else

\[ \pi(r_x) = r_y \]

Example:

(a)  b   c   d   e
Union (a, b)

Find (a) returns b

Union (a, c)

Union (d, e)

Union (e, c)
Proposition.

1. If $x \neq \text{root}$, $\text{rank}(\pi(x)) > \text{rank}(x)$.

2. For any (root) node of rank $k$, there are $\geq 2^k$ nodes in the tree rooted at the node.

Proof idea: Consider the time when the node just got a rank of $k$. This happened when we merge two trees of rank $k-1$. 

\[ \begin{array}{c}
\text{k-1} \\
\text{2 power k-1}
\end{array} \]
\[ 2^{k-1} + 2^{k-1} = 2^k \]

(3) \# nodes of rank \( k \leq n/2^k \).

\[ p \cdot 2^k \leq n \]

\[ \therefore p \leq n/2^k \]

\[ \Rightarrow \max \text{ rank of any node } \leq \lg n. \]

\[ h_t = \lg n. \]

\[ \# \text{ nodes of height } \lg n \leq n/2^{\lg n} \]

\[ = 1 \]

\[ \text{if } t > \lg n \text{ then } \# \text{ nodes } \geq \text{ that} \]
Running time of Kruskal: $O(m \log n)$.

Suppose we had the edges in sorted order for free.

Idea: Path Compression.

$\text{Find}(x)$

If $x \neq \pi(x)$ then

$\pi(x) \leftarrow \text{Find}(\pi(x))$

return $x$
\[
\log^* n \; : \; \# \text{ log operations performed to bring down the value from } \ n \rightarrow 1 \ (\text{or less}).
\]

Suppose \( n = 2 \)

\[
\log n \rightarrow 1024 \quad 10 \quad 4 \rightarrow 2^2
\]

Observation: rank of node may no longer be the bit of the node.
- rank of a node never changes once the node ceases to be a root.

1. If \( x \neq \text{root} \), \( \text{rank}(\pi(x)) > \text{rank}(x) \).

2. For any root node of rank \( k \), there are \( > 2^k \) nodes in the tree rooted
at the node.

(3) # nodes of rank = k ≤ n/2^k.

Smallest rank = 0 , largest rank = \lg n.

Partition ranks as follows:

\{ 1 \}, \{ 2 \}, \{ 3, 2^2 \}, \{ 5, 6, 7, \ldots, 2^{k-1} \}, \{ 17, \ldots, 2^{k} \}, \ldots.

# partition ≤ \lg^* n.

What we are going to do is to calculate the amortized cost of find by considering a sequence of find & union operations starting from an empty data structure.
Each node will have some pocket money.

Total amt of pocket money: $O(n\lg^*n)$

Budget: $O(m\lg^*n)$

Total cost of the seq. $f$ finds must be covered by the budget + pocket money. If we can do this then total cost of $m$ find operations

= $O(m\lg^*n)$.

How much pocket money does each node get?

If rank of a node $\in \{k^i, \ldots, 2k^i\}$
Theorem: The node gets $2^k$.

Total # nodes with ranks belong to
\[ \{ k+1, k+2, \ldots, 2^k \} \]
\[ \leq \frac{N}{2^{k+1}} + \frac{N}{2^{k+2}} + \ldots \]
\[ = \frac{N}{2^k} \left( \frac{1}{2} + \frac{1}{2^2} + \ldots \right) \]
\[ \leq \left[ \frac{N}{2^k} \right]. \]

i. Total amount of dollars to all nodes with rank \( k \in \{ k+1, \ldots, 2^k \} \)
\[ \leq \frac{N}{2^k} \cdot 2^k = n. \]
Total and if pocket money with all nodes \( \leq \lceil n \cdot \lg^* n \rceil \).

When we do find, we make big jumps

\[ x \rightarrow \pi(x) \quad \text{if rank}(\pi(x)) \text{ belongs to a different natural.} \]

Small jumps

\[ x \rightarrow \pi(x) \quad \text{if rank} f \pi(x) \text{ is in the same interval as rank} x. \]

\[ \text{Find } \pi(x) \quad \text{\$1 gets taken from the budget.} \]

Small jump: \$1 gets taken from the pocket money if \( x \).
\[ \text{rank}(w) \in \{k+1, \ldots, 2^k \} \]

\[ O(m^n) \leq 2^k \]

\[ \alpha(n) \]

\[ \pi(w) \]