Cut Property:

For \( G = (V, E) \), consider \( X \subseteq E \) where \( X \subseteq T \) for a MST \( T \).

Take any subset \( S \) of vertices where no edge of \( X \) crosses \( S \leftrightarrow \overline{S} \).

Let \( e^* \) be the edge of \( G \) of min weight that crosses \( S \leftrightarrow \overline{S} \).

Then, \( X \cup e^* \subseteq T \) where \( T \) is a MST.

---

Prim's algorithm:

\( R \) = explored vertices.

\( X \) is a MST on \( R \).

Let \( S = R \) & take the \( e^* \) of min weight out of \( S \).

Say \( e^* = (y, z) \) where \( y \in R, z \notin R \).

Add \( e^* \) to \( X \) and \( z \) to \( R \).

Repeat until \( R = V \).
Kruskal’s algorithm:

Greedy approach:
For input $G = (V, E)$
   Sort $E$ by $\leq$ weight
   Let $X = \emptyset$
   Go thru $E$ in $\leq$ order:
      For edge $e = (y, z)$
         if $X \cup e$ is acyclic
            then add $e$ into $X$

How do we test if $X \cup e$ makes a cycle?

In the graph $(V, X)$, are $y$ & $z$
in the same component?
   if so, then $X \cup e$ makes a cycle
   if in diff. components then adding it is OK.

Use union-find datastructure to check
   if $y$ & $z$ are in the same
   or different components.
Why is Kruskal's algorithm correct?

Let $G = (V, X)$.

Let $c(y)$ be the vertices in the component containing $y$ in $G$.

Let $S = c(y)$.

Note: $e = (y, z)$ is the min weight edge of $G$ crossing $S \cup \overline{S}$.

Why? Suppose $e' = (a, b)$ has $a \in S$, $b \in S$ & $w(e') < w(e)$.

Then $e'$ is considered by Kruskal's alg. before $e$ & it will get added into $X$ since $a$ & $b$ are in dif' f. components. But then $b \in S$ which is a contradiction.

Hence, by the cut property adding $e$ to $X$ we're still on our way to a MST.
Union-find data structure:

- Collection of sets - each set corresponds to a component in the graph \((V, E)\)
- Each set has a unique name - the name will be the root vertex.

Operations:
- Makeset \((x)\): create a new set just containing \(x\)
- Find \((x)\): what is the name of the set containing \(x\)?
- Union \((x, y)\): merge the sets containing \(x\) & \(y\).
  - \(O(\log n)\) time per Find, Union
  - \(O(1)\) time per Makeset

To check if \(x\) & \(z\) are in the same or different components, we just check:

\[
i = \text{Find}(y) = \text{Find}(z)\]

When adding \(e = (y, z)\) into \(X\),
- Do Union \((y, z)\) to merge their components.
Kruskal \((G, w)\):

input: connected, undirected \(G = (V, E)\) with edge weights \(w(e) \geq 0\) for \(e \in E\)

output: MST defined by \(X \subseteq E\)

for all \(v \in V\), MakeSet\((v)\)

\(X = \emptyset\)

Sort \(E\) by \(\uparrow\) weight.

Go through edges by \(\uparrow\) weight:

for \(e = (y, z)\)

if \(\text{find}(y) \neq \text{find}(z)\)

then \(\left[\text{add } e \text{ to } X\right]\)

\(\text{Union}(y, z)\)

Return\((X)\)

Running time: \(O(m \log n)\) time.
Union-find data structure:

Each set is a directed tree:
- edges point up to the root
- name of the set is the root.

Example: \{B, E, J, \} \{A, C, D, F, G, H, I, J\}

Each node \( x \) has 2 values:

1) \( \pi(x) = \text{parent of } x \)
   - if \( x \) is the root
     - then \( \pi(x) = x \)

2) \( \text{rank}(x) = \text{height of subtree below } x \).
Makenset (x):
\[ \pi(x) = x \]
\[ \text{rank}(x) = 0 \]

Find (x):
While \( x \neq \pi(x) \) Do:
\[ x = \pi(x) \]
Return (x)

To merge sets containing x & y, point root of one to root of other.

Key: to minimize depth, point root with smaller depth to larger depth.
So root with smaller rank points to larger rank.
Union \((x, y)\):

\[
\begin{align*}
a &= \text{Find}(x) \\
b &= \text{Find}(y)
\end{align*}
\]

if \(\text{rank}(a) > \text{rank}(b)\)

then \(\Pi(b) = a\)

if \(\text{rank}(b) > \text{rank}(a)\)

then \(\Pi(a) = b\)

if \(\text{rank}(a) = \text{rank}(b)\)

then \(\begin{cases} \Pi(b) = a \\ \text{Rank}(a)++ \end{cases}\)
Key claim: max depth is $\leq \log n$

Hence, find & union take $O(\log n)$ time.

Claim 2: Root of rank $k$ has $\geq 2^k$ nodes in its subtree (including itself).

Proof: By induction on $k$.

Base case: $k=0$: $2^0=1$ & it includes itself.
Assume it's true for rank $<k$.
Consider node of rank $k$.
It was formed by union of $a$ & $b$ of rank $k-1$.
Both $a$ & $b$ have $\geq 2^{k-1}$ nodes in their subtrees.
So $\geq 2 \times 2^{k-1} = 2^k$ in the union.

Let $l = \# \text{ of nodes of rank } k$.

By claim 2, $l \times 2^k \leq n$

$$l \leq \frac{n}{2^k}$$

Let $l=(\log n) + 1$

Then, $l \leq \frac{1}{2} < 1$ so $0$ nodes of rank $>\log n$.