Kruskal's MST Algorithm:

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

Why is it correct?

Cut Property: For $G=(V,E)$,
consider $X \subseteq E$ where $XCT$ for a MST $T$.
Take any SCV where no edge of $X$ crosses $S \rightarrow \overline{S}$.
Let $e^*$ be the min wt. edge in $E$ crossing $S \rightarrow \overline{S}$.
Then, $X \cup e^*CT$ for a MST $T$. 
For Kruskal's, assume current \( X \cup C \) for a MST.
Suppose we're adding edge \( e = (y, z) \).
That means in \( X \), \( y \) & \( z \) are disconnected.
Let \( c(y) \) be \( y \)'s component,
& \( c(z) \) be \( z \)'s component.
Let \( S = c(y) \).
Note, no edge of \( X \) crosses \( S \).
otherwise \( c(y) \) would be bigger.
And \( e \) is the min wt. edge of \( E \)
crossing \( S \).
otherwise we would have considered that lighter edge earlier & added it to \( X \) and expanded \( c(y) \).
Thus by the cut property,
\( X \cup e \cup C \) for 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 which is the root of its "tree".

Operations:

- `MakeSet(x)`: create a new set just containing \(x\).
- `Find(x)`: return name of set containing \(x\).
- `Union(x, y)`: Merge sets containing \(x\) and \(y\).

\(O(1)\) time

\(O(\log n)\) time
Kruskal \((G, w)\)

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

\(X = \emptyset\)

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

For edge \(e = (y, z)\): (go through \(E\) in \(\uparrow\) order)

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

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


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

\(\text{Return } (X)\)

---

Run time:

Sorting \(E\) \(\Rightarrow \) \(O(m \log n)\) time.

\(\wedge\) \(\text{MakeSets} \Rightarrow \) \(O(n)\)

\(O(m)\) \(\text{finds} \& \text{Unions} \Rightarrow \) \(O(m \log n)\)

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

Each set is a directed tree:
- edges point up to the root
- name of the set is the root
Every node also has its rank

\[ \text{rank}(x) = \text{height of subtree below } x \]
\[ \pi(x) = \text{parent of } x \]
if \( \pi(x) = x \) then \( x \) is the root.

Example: \( \{A,B,E,G,\pi\} \)

Diagram:

\[ \text{Diagram with tree structure showing nodes } A, B, C, D, E, F, G, H, I, J \]
MakeSet(x):
\[ \Pi(x) = x \]
\[ \text{rank}(x) = 0 \]

Find(x):
While \( x \neq \Pi(x) \) do:
\[ x = \Pi(x) \]

Return(x)

To merge 2 sets, point root with smaller depth to other root
\[ \Rightarrow \text{minimizes max depth.} \]

Union(x, y):
\[ r_x = \text{find}(x) \]
\[ r_y = \text{find}(y) \]
if \( \text{rank}(r_x) > \text{rank}(r_y) \) then
\[ \Pi(r_y) = r_x \]
if \( \text{rank}(r_y) > \text{rank}(r_x) \) then
\[ \Pi(r_x) = r_y \]
if \( \text{rank}(r_x) = \text{rank}(r_y) \) then
\[ \Pi(r_x) = r_y \]
\[ \text{rank}(r_y)++ \]
Key claim: max depth is \( \leq \log n \).

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

Proof of claim 2:

**Base case:** \( k=0 \): count node itself so \( 2^0 = 1 \) ✓

Assume true for nodes of rank \( \leq k-1 \).

To get node of rank \( k \), merge 2 nodes of rank \( k-1 \)

- each has \( \geq 2^{k-1} \) nodes by induction
- so new subtree has \( \geq 2 \times 2^{k-1} = 2^k \)

Proof of key claim:

Let \( l \) be # of nodes of rank \( k \).

Then \( (l)(2^k) \leq n \)

So \( l \leq \frac{n}{2^k} \)

Let \( k = \log_2 n + 1 \), then \( l \leq \frac{1}{2} < 1 \) so no nodes of rank \( \log_2 n + 1 \).