Minimum Spanning Tree

Kruskal's method: allow a forest of trees.

Prim's: maintain a connected subtree at all time.

1) sort edge by weight (break ties alphabetically)
2) repeatedly pick the next smallest edge, as long as no cycle

$O(|E| \log |E|)$ time

edge with the least weight across a cut that results $T_i$ is part of some MST

$O(|E| \log |V|)$

$O(|V|)$

$O(1)$

$O(|E| \log |V|)$

$O(|V| \log^2 |V|) = \Theta(|V| \log^2 |V|)$

Union-find data structure

Kruskal's has a set of connected components/trees on a given edge

$O(\log |V|) \text{ time}$

merge: if the two node are in different components, then merge the two components

$O(\log^2 |V|)$ repeated by

for all practical matter:

Kruskal: $O(|E| \log |V|)$
for all practical purpose
is a constant

Kruskal: \( O(1 + |V| \log |V|) \)

\[ \begin{align*}
\text{if root } x \text{ and } y \text{ have the same root,} \\
\text{merge: } A - C \\
\text{find}(A) \rightarrow A \\
\text{find}(C) \rightarrow C \\
\text{roots are different} \\
\Rightarrow A, B, \text{ and } C \text{ are in different components} \\
\text{merge}(A, C): \text{ merge the two } A \text{ and } C \\
\text{after merge, } A \text{ and } C \text{ should have a common root} \\
\text{adjust the rank of root} \\
\end{align*} \]

root check:
\( \pi(x) \): parent of \( x \)
\( y, \pi(x) = x \text{ then root} \)
\( \text{find}(x) = \text{find } x \text{'s root} \)
\( \Rightarrow \text{ follow the chaining pointers} \)
\( \text{while } x \neq \pi(x) \)
\( x = \pi(x) \)

edge: \( C - D \)
\( \text{find}(C) = A \)
\( \text{find}(D) = D \)
\( \text{move } A \text{ and } D \)}
**Merge**:

1. If root x & y have the same rank, then rank(x) = rank(y) + 1.
   - Set r(x) = x

2. Point the root with smaller rank to the root with higher rank.

**Edge**: A – D

We have:

- $\text{find}(A) = A$
- $\text{find}(D) = A$

$\Rightarrow$ Rejeve the edge.

**Edge**: A – B

We have:

- $\text{find}(A) = A$
- $\text{find}(B) = B$

We merge (A, B)

**Edge**: C – F

We have:

- $\text{find}(C) = A$
- $\text{find}(F) = C$

We merge (A, C)

- $O(1)$ time operation

---

**Property 1**: Except for the root node, $\forall x, \text{rank}(x) < \text{rank}(r(x))$ by construction.

**Property 2**: If a node has rank $r_j$, then it has $\geq 2^j$ nodes in the subtree. Proof by induction.
Proof by induction

Base case: \( r = 0 \)

\[ S^0 \]

\[ \begin{array}{c}
\text{A} \\
\text{B}
\end{array} \]

\[ z^0 = 1 \quad 2^0 = 1 \]

\[ r = \frac{1}{2} \]

\[ \frac{1}{2} \]

\[ \frac{1}{2} \]

\[ 2^r = 2^0 = 2 \]

Maximum rank is \( \log(|V|) \)

- Each component has rank 0
- It has all the node \(|V|\) in G

\[ 2^r = |V| \]

\[ r = \log(|V|) \]

\[ \Rightarrow \text{find operations take } O(\log(|V|)) \text{ in the worst case} \]

\[ \Rightarrow \text{bring down even more} \]

\[ O(\log^*(|V|)) \text{ per operation} \]
$$G = (V, E)$$

$$|V| = 2^{\log_2 26}$$

$$\log_2 (|V|) = \log_2 (2^{14}) = 14$$

$$\log_2 (2^{14}) = 14 = 2^7$$

$$\log_2 (2^7) = 7 = 2^5$$

$$\log_2 (2^5) = 5 = 2^3$$

$$\log_2 (2^3) = 3 = 2^1$$

$$\log_2 (2) = 1$$

Amortized costs

even though a few operations can take longer

if we consider a sequence of such operations, we can bound the time within the given $$O(\cdot)$$

$$O(\log^* |V|)$$ per operation

$$O(\cdot)$$

Amortized cost $$O(\log^* |V|)$$ for union find

$$2 \times |E|$$ find operations in Kruskal's

Amortized analysis
via pack compression ← pretty much free!

\[ \text{find}(x) = A \quad \text{move} \]

during find from x
change all parent pointers on that path to point to root

Set \( \alpha \) ranks for the node in union-find data structure:

\[
\begin{align*}
g_1, g_2, g_3, g_4, \ldots, g_n \\
\{ 1 \}, \{ 2 \}, \{ 2^1 \}, \ldots, 2^{k-1} \} \\
\{ 2^k \}, \ldots, 2^{k+1} \} \\
\{ 2^{k+1} \}, \ldots, 2^{k+2} \} \\
\{ 2^{k+2} \}, \ldots, 2^{k+3} \} \\
\vdots \\
\{ 2^{k+n} \}
\end{align*}
\]
Any node $x$ will get one budget the first time it ceases to be a root.

A node at swap $k$ gets a budget $2^k$.

It can use this budget to adjust the parent pointers.

Total budget does not exceed $\log^* |V|$ per node.

$|V| \log^* |V|$ total budget given across all nodes.

**Property 2**: at most $\frac{n}{2^y}$ nodes of rank $r$.

A node with rank $r$ has $\geq 2^r$ nodes under it.

The number of nodes at rank $r \equiv n_r$.

$$n_r \cdot 2^r \leq n$$

$$n_r \leq \frac{n}{2^r}$$

$Y \geq \log \log n$.
If we give each node in a graph a weight of 2

Each step over all nodes

\[
\left\{ \begin{array}{l}
\frac{1}{x} + \frac{1}{y} + \frac{1}{z} + \cdots \\
\frac{1}{x} + \frac{1}{y} + \frac{1}{z} + \cdots \\
\end{array} \right.
\]

\[
\frac{2^0 + 2^{-1} + 2^{-2} + \cdots}{1 + 2^{-1} + 2^{-2} + \cdots}
\]

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

Property 1:

\[
\frac{2^0 + 2^1 + 2^2 + \cdots}{1 + 2 + 2^2 + \cdots}
\]

or more nodes with rank \( \nu \).