1. Let $T$ be a tree whose longest path is length $k$ ($k$ edges). Let $P_1, P_2$ be two different paths in $T$, both of length $k$. Show that $P_1$ and $P_2$ must have a vertex in common.

![Diagram of tree with paths $P_1$ and $P_2$.]

If $P_1$, $P_2$ disjoint, there’s path from vertex $v$ on $P_1$ to vertex $w$ on $P_2$ of length $\geq 1$.

Eusing that tree is connected?

By taking shortest possible such path, can assume that it only intersects $P_1$ at $v$, $P_2$ at $w$.

Assume $w$ has that distance $d(v,w) > d(v,u)$, $d(w,v) > d(w,v)$.

Then $b-v-w-d$ is path of length $\geq \frac{k}{2} + 1 + \frac{k}{2} > k$.

We've found a shorter path, contradiction.

Name: SOLUTIONS
2. A tree on vertex set \( \{1, \ldots, n\} \) has Prüfer code \( nnn\ldots n \) (\( n - 2 \) n’s). Draw the tree in the space below.

\[\begin{array}{c}
\text{n-1 vertices} \\
\text{in a ring,} \\
\text{labeled 1 through n-1}
\end{array}\]

3. How can you tell by looking at the Prüfer code of a tree on vertex set \( \{1, \ldots, n\} \), that the tree is a path? [Without, of course, just building the tree from the code ... you should describe a way that just involves scanning the code.]

\[\text{# occurrences of vertex name in code = degree - 1}\]

In path, \( n-2 \) vertices have degree 2, so \( n-2 \) names appear once.

This accounts for all of code

So tree = Prüfer code has no repeated letters.
4. Let $G$ be a graph (not necessarily connected) on $n$ vertices $\{v_1, \ldots, v_n\}$. Put weights on the edges of the complete graph on vertex set $\{v_1, \ldots, v_n\}$ as follows: if $v_iv_j \in E(G)$, set $w(v_iv_j) = 0$, and if $v_iv_j \notin E(G)$, set $w(v_iv_j) = 1$. Run Kruskal’s algorithm on this weighted complete graph. Explain (with justification!) how you can use the output of Kruskal’s algorithm to deduce the number of components that $G$ has.

Suppose $G$ has $k$ components, $G_1 \ldots G_k$, with $n_1, \ldots, n_k$ vertices, $n_1 + \ldots + n_k = n$.

Kruskal picks out as many edges of weight 0 as possible first (without creating cycle), so $K$ picks out spanning tree inside each of $G_1, \ldots, G_k$ for $(n_1 - 1) + \ldots + (n_k - 1) = n - k$ edges of weight 0.

Remaining $k - 1$ edges (to get up to required $n - 1$) have all weight 1, so Kruskal outputs $\#k - 1$.

Conclusion: $\#$ Components = Output of Kruskal + 1.
5. By using the Prüfer code, find a formula for the number of trees on vertex set \( \{1, \ldots, n\} \) \((n \geq 3)\) that have exactly \( k \) leaves (\( k \) between 2 and \( n-1 \)). Your formula should not involve a summation. [Hint: Think of the Prüfer code as a function \( f: \{1, \ldots, n-2\} \to \{1, \ldots, n\} \), with \( f(i) \) being the number in the \( i \)th position of the code. If the tree has exactly \( k \) leaves, what do you know about the range of the function?]

\[ \binom{n-2}{k} \text{ ways to choose which } h \text{ leaves.} \]

These \( k \) numbers don't appear in Prüfer code, all other numbers do, so Prüfer code is \( f: \{1, \ldots, n-2\} \to \text{set of size } n-h \)

That is \textbf{surjective} \( \checkmark \) \textbf{Shriling # of second kind}.

There are \( (n-h)! \text{ } S(n-2, n-h) \) surjective functions from set of size \( n \) to set of size \( n-h \), so

\( \# \text{ trees with exactly } k \text{ leaves is } \)

\[ \binom{n-2}{k} (n-h)! \text{ } S(n-2, n-h) \]