Cayley's theorem and Prüfer codes

Theorem 1 (Cayley)**.** *The number of spanning trees on n labeled vertices is n^{n-2}.*

Proof (due to Prüfer).

Let us denote the vertices by $1, 2, ..., n$. The key idea is to define a bijective function f between the set of all trees on n labeled vertices and the set of all sequences of length $n-2$ using the symbols $\{1, 2, ..., n\}$. The function f will assign to each tree a unique sequence of length $n-2$. Proving that f is bijective, and using the fact that the cardinality of the set of sequences of length $n-2$ using $\{1, 2, ..., n\}$ is n^{n-2}, gives us that the number of spanning trees on n labeled vertices is n^{n-2}. Therefore, we need to argue that f is a bijection.

We can see this bijection as a pair of algorithms: one transforming every tree on n vertices into a sequence, and the other transforming every sequence into a tree. In what follows, we present these algorithms.

The following algorithm takes a tree as input, and yields a sequence of integers:
Step 1: Find the leaf with the smallest label and write down the number of its neighbor.
Step 2: Delete this leaf, together with the only edge adjacent to it.
Step 3: Repeat until we are left with only two vertices.

Therefore, the image of every tree T on n labeled vertices under the function f will be the output of the algorithm above for the tree T. One can see that the function is well-defined, i.e. for every tree given as an input, the algorithm will output a sequence of length $n-2$. Also, one can prove that the function is injective, that is the algorithm does not output the same sequence for distinct trees.

We present now the algorithm that reconstructs a tree from the Prüfer code:
Step 1: Draw the n nodes of the tree, and label them from 1 to n.
Step 2: Make a list of all the integers $(1, 2, ..., n)$. This will be called the list.
Step 3: If there are two numbers left in the list, connect them with an edge and then stop. Otherwise, continue on to step 4.
Step 4: Find the smallest number in the list which is not in the sequence. Take the first number in the sequence. Add an edge connecting the nodes whose labels correspond to those numbers.
Step 5: Delete the smallest number from the list and the first number in the sequence. This gives a smaller list and a shorter sequence. Then return to step 3.

We can think of this second algorithm as the inverse of the function f. In fact, in order to prove that the function f is bijective, it is enough to prove that the algorithm

of creating a tree from a Prüfer sequence is correct: on one hand, that it does not "get stuck", and, on the other hand, that its output is indeed a tree.

At any step the size of the set of used vertices (used leaves) plus the length of Prüfer sequence is $n-2$, which means that we always have two vertices from the list that we can use as a new leaf

We now prove that the output is a tree by induction on the length of the sequence $S = n-2$. We check for $S = 1$, assume true for $S = k$ and prove it for $S = k+1$. We know that there must exist a vertex of degree one in the list. Consider the vertex of degree one in the list with smallest label and the first vertex in the sequence. The output of the algorithm will be that edge together with the output of the algorithm when given the same sequence without the first element. By the induction hypothesis, we know that this output is a tree, and, by adding one vertex and its emerging edge we obtain again a tree.

Thus, the algorithm does not get stuck, and, the output is a tree.

Also, one has to argue that the encoding procedure applied to T always yields the original Prüfer code, which is equivalent to the fact that the second algorithm is, in some sense, the inverse function of the first algorithm.

In order to prove this, it is enough to verify that, at each step, l_i is always the leaf with smallest label in G_i, where l_i represents the leaf removed at step i by the encoding algorithm, and $G_i = (\{1, 2, \ldots, n\}, \{e_i, e_{i+1}, \ldots, e_{n-1}\})$ is the graph created in step $n-i$ by the decoding algorithm. By the definition of l_i, a smaller leaf could occur only among $\{l_1, \ldots, l_{i-1}\}$ or among $\{p_i, \ldots, p_{n-2}\}$. One can easily see that all the vertices $l_1, \ldots, l_{i-1}$ are isolated in G_i, so it is enough to refer to the second group. Let us consider a vertex p_k, $i \leq k \leq n-2$. In the graph G_k, p_k is a neighbor of the leaf l_k, and it has yet another neighbor since it lies in the single connected component of G_{k+1} with at least 2 vertices, so it cannot be the leaf, which completes the proof.

For a more detailed exposition on this, refer to J. Matoušek, J. Nešetřil, *Invitation to Discrete Mathematics*, chapter 8.4 A proof using the Prüfer code