Double Counting and Counting by Bijection

October 13, 2022

Double counting is the technique of **correctly** counting the same number, in two different ways, and equating the answers to reach a useful conclusion.

Example application **The first theorem of graph theory**.

$$\Sigma_{v \in V} d(v) = 2|E|$$

We showed how do translate from a graph drawing to an adjacency matrix to an incidence matrix.

In he incidence matreix the number of 1's in a row indicates the degree of that vertex. The number of 1's in a column is always two and the number of columns is the number of edges. The left hand side of the above theorem is the total number of 1's calculated row-wise. The right hand side is the total number of 1's counted column-wise. This is double counting and hence we get the theorem.

Counting by bijection

We are going to derive **Cayley's formula** for the number of labelled trees on a set of n nodes labelled with the labels $1, \ldots, n$. These are unrooted trees. The formula for the number of such trees is n^{n-2}.

A **tree** is a connected acyclic graph.

We are going to use the technique of proof by bijection. The core idea of this technique is that for two finite sets: Their cardinalities are identical if and only if there exists a bijective function from one to the other.

We are goig to use the following set whose cardinality is easy to compute. Consider the set of all strings of length k over an alphabet consisting of t letters or symbols. By the rule of product, we know that the size of this set is t^k.

We are going to map the set of all labelled trees over the set of vertices $\{1, \ldots, n\}$ to a string of length $n-2$ over the alphabet $\{1, \ldots, n\}$, i.e. the same as the vertex set. This mapping will be shown to be **bijective**. How

do we argue that the function is bijective? We associate with any tree a unique code and with any code a unique tree. Thus, this proof consists of two algorithms. One takes a tree and produces a string (the prufer code) and the other takes a prufer code and produces a tree.

Algorithm: Tree to code While the number of nodes in the current tree is greater than 2

1. List all the leaves of the current tree
2. Find the lowest indexed leaf
3. Delete this leaf
4. Write down the label of the unique neighbour of the deleted node, in step 3 to the code written so far. Concatenate the label to the right extreme of the string.

Loop

We get the prufer code in the end.

The degree of a node in the tree is the number of times it appears in the prufer code -1.

List all the leaves of the tree (those that do not appear in the prufer code. Write down the elimination order of the leaves based on this. In the first step, it will be the lowest indexed leaf. Once a leaf is eliminated, it wont be present in the residual subtree. its unique neighbour would be the symbol written down at that round. Reduce the degree of that symbol by 1 in the reduced tree. If it becomes a leaf (degree 1) add it to the new leaf set. **Algorithm: Prufer code to labelled tree**

1. Check the length l of the code.
2. Add 2 to the answer of part 1 to get the number of nodes in the tree.
3. Ensure all the symbols used in the prufer code are from the set $1, \ldots, l+2$, else it is an illegal prufer code.
4. Draw n distinct points for the vertices of the tree and label them 1 to n.
5. The algorithm to construct the code from tree always stops when there are two nodes left. One of the nodes left is always the highest indexed node in the whole tree.

The prufer code we got for the tree drawn in the class was $1, 4, 4, 4, 6$. Let us show how to reconstruct the tree. The code is of length 5 and all symbols are from the numbers 1 to 7 which is therefore a legal prufer code.

Let us look at the elimination order of leaves when the prufer code was constructed. The original leaves set is those appearing zero times in the prufer code: $\{2, 3, 5, 7\}$ Thus in the first step we would have eliminated the node 2 and we wrote 1, thus 2 is adjacent to 1. To list the whole tree on 7 nodes, we need to list six edges (every tree on seven nodes has six edges). $E = \{(1, 2)\}$ is the current listing. Node 2 has been eliminates. The reduced prufer code is $4, 4, 4, 6$ as we have already accounted for the symbol 1. Now notice that 1 is not in the reduced prufer code and node 2 is out. Thus the lowest indexed leaf in the new tree is 1. We have written 4 next. Thus 1 is adjacent to 4. $E = \{(1, 2).(1, 4)\}$. Node 1 is eliminated and the reduced string is $4, 4, 6$. There are no new leaves created here so the lowest indexed leaf is 3 (nodes 1 and 2 have been eliminated). And the symbol is 4. Thus we must have deleted node 3 which was adjacent to node 4. $E = \{(1, 2), (1, 4), (3, 4)\}$. The reduced rufer code is $4, 6$. No new leaf has appeared because no symbol entirely disappeared in the current round of the prufer code. Thus the smallest indexed leaf is 5 (1,2,3 have been eliminated earlier) and we wrote they symbol 4, thus the node 5 was adjacent to node 4. Thus $E = \{(1, 2), (1, 4), (3, 4), (5, 4)\}$. The reduced prufer code is now 6. In the current round the symbol 4 completely disappeared from the prufer code. Thus it is now a leaf and is the lowest indexed remaining leaf (1,2,3 of lower indices have been eliminated. Even 5 has been eliminated, but it is of higher index than 4 so it is irrelevant). Thus The remaining leaves are 4,7. 4 is eliminated and we wrote 6 thus, its unique neighbour was 6. The new edge set is $E = \{(1, 2), (1, 4), (3, 4), (5, 4), (4, 6)\}$. This leaves two nodes remaining in the end which are 6,7 and they form the edge $(6, 7)$. Thus the original tree is $V(T) = \{1, 2, 3, 4, 5, 6, 7\}$ and $E(T) = \{((1, 2), (1, 4), (3, 4), (5, 4), (4, 6), (6, 7)\}$. Students who were present and made notes, please cross check that this is consistent with the tree we drew on the board. Also, I remembered the prufer code, hopefully correctly, as $1, 4, 4, 4, 6$. This explains the reverse direction. Thus, by the technique of **counting by bijection**, the number of labelled trees on n nodes is n^{n-2}.