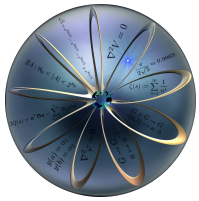


planetmath.org

Math for the people, by the people.

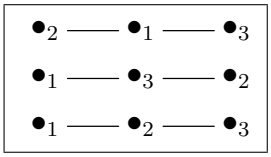
number of unrooted labeled trees

Canonical name	NumberOfUnrootedLabeledTrees
Date of creation	2013-03-22 17:31:20
Last modified on	2013-03-22 17:31:20
Owner	rm50 (10146)
Last modified by	rm50 (10146)
Numerical id	6
Author	rm50 (10146)
Entry type	Theorem
Classification	msc 05C30
Defines	Prufer code
Defines	Prüfer code

Theorem 1. *(Cayley) The number of [isomorphism classes of] labeled trees on n vertices $[n] = \{1, 2, \ldots, n\}$ is n^{n-2}.*

This is sequence A000272 in the `http://www.research.att.com/ njas/sequences`Online Encyclopedia of Integer Sequences

For $n = 1$ and $n = 2$ the result is obvious. For $n = 3$, the possible trees are



while for $n = 4$, the $4^2 = 16$ trees fall into two groups: 12 trees with linear structure and 4 with a star structure. There are 12 linear trees since there are 24 orderings of $1, 2, 3, 4$ and mirror orderings give the same tree; there are 4 star trees since the tree is determined by its central element.

There are many proofs of this theorem; the demonstration we give uses the *Prüfer bijection*, which associates with each such tree its *Prüfer code*; it is easily seen that the number of Prüfer codes on $[n]$ is n^{n-2}.

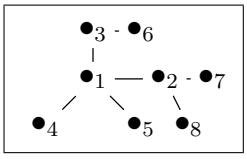
A *Prüfer code* on $[n]$ is a sequence of $n - 2$ elements $a_1, a_2, \ldots, a_{n-2}$ chosen from $[n]$ (repetitions are allowed). Obviously there are n^{n-2} codes on n symbols.

Given a tree T on $[n]$ with $n \geq 3$, convert it to a Prüfer code using the following algorithm:

```
for j = 1 to n − 2
  let l_j be the vertex in T with the smallest label
  let a_j be the vertex adjacent to l_j
  remove l_j from T
next j
```

When this process completes, the result will be a Prüfer code (of length $n - 2$) on n symbols; this mapping is clearly well-defined.

Let's walk through a more complicated example. Consider the following tree on 8 vertices:



Using the algorithm above, we get

l_j	a_j
4	1
5	1
6	3
3	1
1	2
7	2

and thus the Prüfer code associated with this tree is $1, 1, 3, 1, 2, 2$.

The easiest way to see that this map is a bijection is by explicitly constructing its inverse. To do this, we must show how to construct a tree T on $[n]$ given a Prüfer code on n symbols.

Note first that the code for a given tree contains only the non-leaf vertices, and that each non-leaf vertex appears at least once in the code. The first of these statements is obvious from the algorithm - given that $n \geq 3$ and since the algorithm stops when there are only two vertices left in the tree, no leaf vertex in the original tree can be selected as an a_j.

To see that every non-leaf must appear in the list, note that each vertex was either removed as one of the l_j or it remains in the two-vertex tree at the end of the process. In either event, one of its adjacent vertices was removed from the tree at some point, so the vertex would appear in the code. In fact, it is clear that the number of times each vertex appears in the code is one less than its degree in the tree.

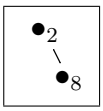
Given this, it's rather straightforward to reconstruct the tree. Consider a_1. The leaf that was removed (l_1) must be the smallest leaf vertex, which is the smallest number not appearing in the code. It must attach to a_1. Essentially, this process continues as one moves forward through the a_j, except that allowances must be made for non-leaf vertices that are removed at later steps as they become leaf vertices. Here is an algorithm to reconstruct a tree from a Prüfer code:

```
A ← the set of leaf vertices
T ← the connected tree on the two vertices a_{n-2} and the highest-numbered
leaf
for j = 1 to n − 2
   add a_j and the lowest-numbered node l in A to T (if not already
in T)
   add an edge in T between a_j and l
```

remove l from A
if $a_k \neq a_j$ for any $k > j$, add a_j to A
next j

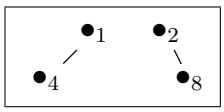
Let's see how this process works on the Prüfer code above.

We start by letting T be the tree on the two vertices 2 and 8 with an edge between them:

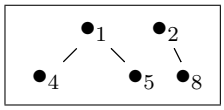


The initial value for A (the available vertices) is $4, 5, 6, 7$.

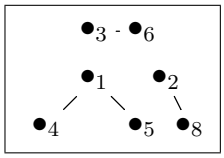
$a_1 = 1$, so add $1, 4$, and an edge between 1 and 4 to T. 1 appears later in A, so we do not add it to A.



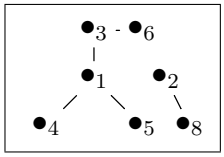
$a_2 = 1$ and $A = \{5, 6, 7\}$, so add 5 and an edge between 1 and 5:



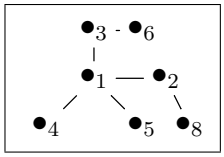
$a_3 = 3$ and $A = \{6, 7\}$. Add $3, 6$ and an edge from 3 to 6. This is the last occurrence of 3 in the code, so add 3 to A.



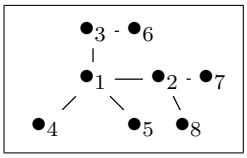
$a_4 = 1$ and $A = \{3, 7\}$. Add an edge from 1 to 3. Add 1 to A.



$a_5 = 2$ and $A = \{1, 7\}$. Add an edge from 1 to 2.



$a_6 = 2$ and $A = \{7\}$. Add 7 and an edge from 2 to 7.



We have reconstructed the original tree.