Discrete Mathematics 03/04/2019

Lecture 17: Trees

Instructor: Goutam Paul *Scribe: Mohammad Anees Parwez*

Contents

17.1 Introduction

In this lecture we would emphasis on the hierarchical relationships between the individual elements or nodes of a graph. We develop discrete structure, called *trees* which can model these relationships. One might be interested in finding the shortest point between two vertices in the given graph. In these class of 'shortest distance' problems, the abstract essence of hierarchy is carried in the 'length' of path.

Trees find its application as early as the 1850s when the English mathematician Arthur Cayley tried to enumerate the structural isomers of alkane. Contemporarily, trees are used extensively in computer science. From locating particular element in a list to developing winning strategies in chess to analyze syntactic structure of sentences, trees have utilization in a wide variety of algorithm.

17.2 Acyclic Graphs and Trees

Definition 17.2.1. *A graph G=(V,E) is* ***acyclic*** *if it has no cycles.*

Linguistically, trees suggest branching out from a root and never completing a cycle. A same quintessence is carried in the formal definition.

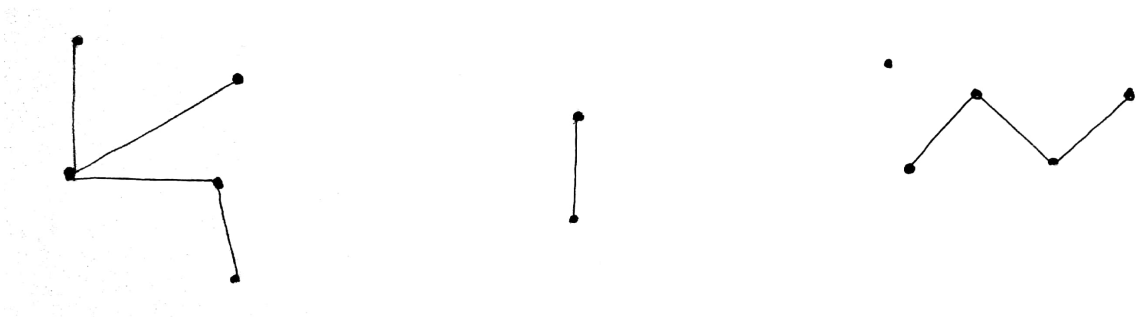

Figure 1: Acyclic graphs

Definition 17.2.2. *A **tree** is a connected acyclic graph.*

Definition 17.2.3. *In a tree, a vertex with degree 1 is called a **pendent vertex** or a **leaf**.*

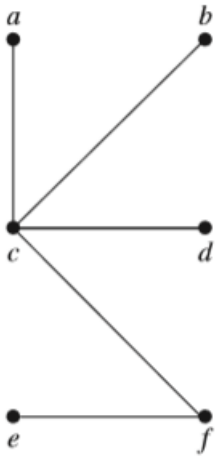

Figure 2: Tree

Figure 2 is a tree because there is no isolated vertex.It is connected acyclic graph. Further note that vertices a,b,d,e each have degree one, hence each of these vertices are *pendent vertex*.

In questions such as deciding weather a given graph is a tree or otherwise, the generic definition as above may prove too hefty. It turns out that trees have many equivalent characterization, any of which could be used to define a tree. Before moving on to these equivalent

definitions we prove that 'deletion a leaf from a tree produces a smaller tree'.

Theorem 17.2.4. *Every tree with at least two vertices has at least two leaves. Deleting a leaf from a tree with n vertices produces a tree with n-1 vertices.*

Proof. A connected graph with at least two vertices has an edge. In an acyclic graph, an endpoint of maximal non-trivial path has no neighbour other than its neighbour on the path. Hence, the endpoint of such a path is a leaf. This proves the first statement of the lemma.

Let v be one of leaf of a tree G and let G'=G -v. We also remove the now isolated vertex v from G'. A vertex of degree 1 belongs to no path connecting two other vertices. Therefore, for $u,w \, \epsilon V(G')$ then every u-w path in G is also in G'. Hence G' is connected. Since deleting a vertex cannot create a cycle, G' also is acyclic. Together we have that G' is a tree with *n-1* vertices.

□

17.3 Characteristic properties *of* Trees

We are now ready to take up equivalent definitions of trees. Our proofs for equivalence uses induction, prior result and other common prov technique.

Theorem 13.3.1 The following statements are equivalent:
(a) G is a connected acylic graph with n vertices.
(b) G is a connected graph with n vertices and n-1 edges.
(c) G is an acyclic graph with n vertices and n-1 edges.
(d) There is exactly one u-v path between any two different vertices u and v of G.

Proof. We note that the equivalence of first three statement is same as establishing that any two of {connected, acylic, *n-1* edges} together imply the third.

(a)⇒ {(b),(c)}

This is same as proving that (acylic,connected) together imply *n-1*

edges.
We use induction on n. For $n = 1$, a graph with 1 vertex is trivially acylic and connected. Also it has no edge. For $n > 1$ suppose the implication holds for less than n vertices. For a n-verticed acyclic connected graph G, Theorem 13.2.4 provides provides a leaf and states that G'=G - v (also removing vertex v) is connected acyclic. The induction hypothesis states that G' has $n-2$ edges. Now in G, only one edge is incident on v. Thus moving from G' to G increases only one edge. Hence, we conclude G has $n-1$ edges.

(b)⇒{(a),(c)} equivalently (connected, $n-1$ edges) together imply acyclic.
Delete edges one by one until the resulting graph G' is acyclic. Now G' is a acyclic connected graph, the previous paragraph establishes that it has $n-1$ edges. But G has $n-1$ edges too. So G' can be obtained from G without removing any edge from G , thus G itself is acylic.

(c)⇒{(a),(b)} equivalently (acylic, $n-1$ edges) together imply connected.
Let $G_1, G_2, ...G_k$ be components of G. Since, every vertex appears in one component $\sum_{i=0}^{k} (n(G_i)) = n$. Since G has no cycles, each component satisfies (a). Thus for each component $e(G_i) = n(G_i) - 1$. Summing over edges we get $e(G) = n - k$. But $e(G) = n - 1$. and hence, $k = 1$ or there is only one component in G or G is connected.

(a)⇒(d) . Since G is connected each pair of vertices u,v is connected by a path. If some pair is connected by two distinct path, we can choose the shortest pair P,Q of distinct path with same endpoint. This implies $P \cup Q$ is a cycle, which contradicts the hypothesis of (a). Thus there is a unique path connecting given two vertices u,v.

(d)⇒(a) if for pair $u, v \in V(G)$ there exist a path, then G is connected. If we suppose that G is cyclic then there exist more than one $u - v$ path for some $u, v \in V(G)$. However this contradicts the hypothesis of d. thus G is acylic too. □

17.4 Distance

In the introductory paragraph we hinted towards solving 'minimum distance' problems with the use of trees. Having equipped with the tools developed above, we begin exploration in this direction.

Definition 17.4.1. *Given a graph G=(V,E). With every pair of vertices* $(u, v), \{u, v\} \in V(G)$ *we associate* ***d(u,v)*** *which is defined as minimum path-length between u and v. The* ***path-length*** *of path is the number of edges along that path.*

Note that the concept of distances doesn't limit to connected graphs. If $u,v \in V(G)$ for some graph G, $d(u, v)$ is set to infinity in case u and v are not connected.

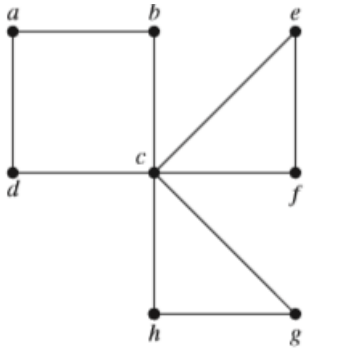

Figure 3

In Figure 3, all vertices adjacent to each other have d(.,.)=1.
d(a,c)=2 (a-b-c) d(b,e)=2 (b-c-e) etc.
d(a,e)=3 (a-b-c-e) d(a,h)=3 (a-b-c-h) etc.

Theorem 17.4.2. *d(u,v) is a metric.*

Proof. To prove that lemma we need to establish:
(i)$d(u, v) \geq 0$,equality prevails only if u=v;
(ii)$d(u, v) = d(v, u)$;
(iii)$d(u, w) + d(w, v) \geq d(u, v)$.

The validity of (i) is trivial. Since we are working under un-directed graphs, (ii) follows.

Let P be the path of minimum path-length between u,v. If P passes through w then equality prevails. In case P doesn't pass through w then we need to show that $d(u, v)$ is smaller. We suppose for the sake of contradiction that $d(u, v) > d(u, w) + d(w, v)$. This shows that the length of minimum path P between u and v, is larger than the length of path $u - v$ path passing through w. This violates that P is minimum path-length. Hence, $d(u, v) \leq d(u, w) + d(w, v)$ □

We next prove an important result which provides an upper bond on sum of minimum path-lengths from a fixed vertex to all other vertices. The puissance of the result can be realized in computer science where complexity of algorithm incorporating tree searches need to be adjudged.

Theorem 17.4.3. *Let u be any fixed vertex in a tree T with n vertices. Then* $\sum_{v \in T} d(u, v) \leq \binom{n}{2}$

Proof. We proof the result using strong induction principle. For the base case $n = 2$. This has only one edge, also $\binom{2}{2} = 1$; thus LHS and RHS match.
We assume the result holds for less than n vertices.
Now consider a tree T with n many vertices. We fix the vertex u. Let $deg(u) = k$; consider the graph G obtained by removing this vertex u and all edges ending in u. Then G has *n-1* vertices.

Claim: G has k components.
Pf: The tree T is connected and components of G are isolated from each other; thus from each component there must be vertex which shares a common edge with u. It follows that number of components are same as number of edges incident on u which is $deg(u) = k$.

The k components of G are $G_1, G_2, ...G_k$. In each component $\exists$ unique vertices $v_1, v_2, ...v_k$ which share an edge with u. Let n_i denote number of vertices in G_i. Consider G_1. $\forall v \, \epsilon \, G_1$

$$d(u, v) \leq d(u, v_1) + d(v_1, v)$$

$$\sum_{v \in G_1} d(u, v) \leq \sum_{v \in G_1} d(u, v_1) + \sum_{v \in G_1} d(v_1, v)$$

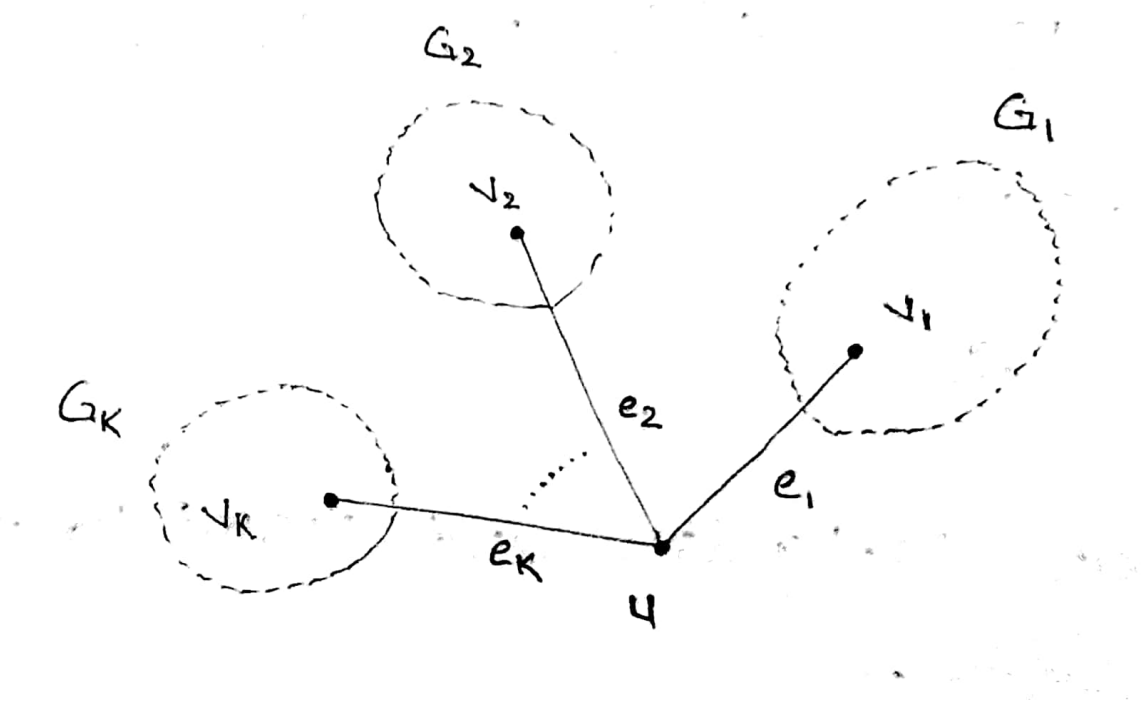

Figure 4: Proof demonstration

Since u and v_1 are adjacent, the first term on RHS equals n_1. Each of the components of G are connected. Moreover since T is acyclic, so are the individual components of G. Thus each component of G is a tree with less than n vertex. Principle of strong induction applies and implies $\sum_{v \in G_1} d(v_1, v) \leq \binom{n_1}{2}$. We do this for all k components and sum up the results.

$$\sum_{i=1}^{k} \sum_{v \in G_i} d(u, v) \leq \sum_{i=1}^{k} n_i + \sum_{i=1}^{k} \binom{n_i}{2}$$

$$\implies \sum_{v \in T} d(u, v) \leq (n-1) + \binom{n-1}{2} = \binom{n}{2}$$

And hence, the result stands proved!

□

17.5 Rooted Trees

In some situations we may be interested in considering one of the vertices of trees as our node. Once we fix a node, we can not only arrange other vertices in order of their distance from this fixed node but also introduce essence of direction to each edge as follows: Because there is a unique path from the root to each vertex of the graph(Theorem 13.3.1 above), we direct each edge away from the root. Thus, a tree together with its root produces a directed graph called a *rooted tree*.

Definition 17.5.1. *A **rooted tree** is a tree where one vertex is taken as root and vertices are arranged according to their distance from the root.*

We can change an un-rooted tree into a rooted tree by choosing *any* vertex as the root. Note that different choices of the root produce different rooted trees.

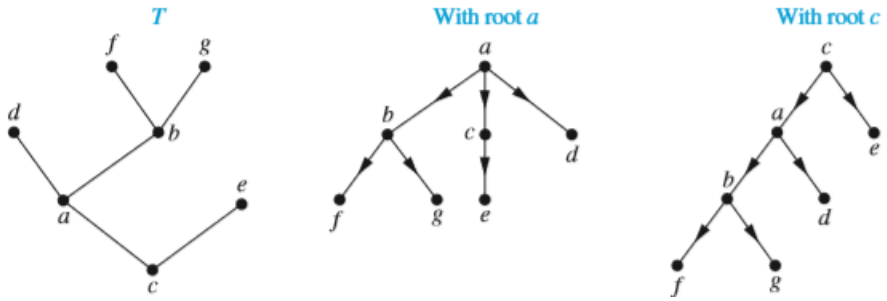

Figure 5: Figure on left is given un-rooted tree T, the middle figure made with designating vertex a as root, and rightmost figure is made by designating c as root

Note that in a rooted tree all the vertices in first generation as at one unit path-length from the root node. All the vertices in second generation are at two unit path-length from root node and so on.

The terminology for *trees* has botanical and genealogical origins. Suppose that T is a rooted tree.If v is a vertex in T other than the root, the **parent** of v is the unique vertex u such that there is a directed edge from u to v. Bearing to the familial similarity we can define similar relationships between the vertices of a rooted tree.

When u is the parent of v, v is called a **child** of u.

Vertices with the same parent are called **siblings**.

The **ancestors** of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.

The **descendants** of a vertex v are those vertices that have v as an ancestor.

A vertex of a rooted tree is called a **leaf** if it has no children.

In the discussion so far, we haven't yet discussed the concept when the children are chronologically or otherwise ordered. Familial trees can be drawn in a manner where in a given generation the youngest child is to the left extreme and the oldest child (of same generation) is to the right extreme. Borrowing this abstraction of time or weightage to different nodes of same generation, we introduce *Ordered Rooted trees.*

Definition 17.5.2. *An **Ordered Rooted tree** is a rooted tree where the children of each internal vertex are ordered.*

Ordered rooted trees are drawn so that the children of each vertex are shown in order from left to right.
We call a tree as **binary tree** if each vertex has no more than two children. In an ordered binary tree (often referred as binary tree itself) the first child is called the **left child** and the second child is called the **right child**.

Problem: Count the number of ordered binary trees having n vertices.

Solution: We try to approach the question bearing in mind the recursive nature of trees. Choose any one of the n vertices, fix it as root. Now, we are left with *n-1* vertices. Fix an integer k s.t. $0 \leq k \leq n-1$. Now, we move k-many points to the left the rest *n-1-k* many points to the right.Let τ_n denote number of such ordered binary trees with n vertices. The number of ordered binary trees (having n vertices) with k-many vertices to left is given by $\tau_k \ \tau_{n-1-k}$

Different choices of k would give different tree structure (as the tree is ordered). So, we need to sum over all such choices of k. Hence,

$$\tau_n = \sum_{k=0}^{n-1} \tau_k \, \tau_{n-k}$$

The above can be solved by the method of generating functions. Recalling from earlier chapter on 'Generating functions', we deduce τ_n is the n^{th} Catalan number.

$$\tau_n = \frac{\binom{2n}{n}}{n+1}$$

□

We next, present a similar problem on cardinality of the set of *labeled* trees with n vertices. The term **labeled** is used to highlight that vertices of graph are distinguishable.

For example, in the figure below had the labels not been present, all three graphs would have represented the same structure. However, as the vertices are labeled; each graph represents a different connectivity structure.

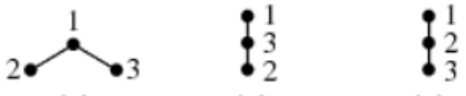

Figure 6: Three distinct labeled graphs

The result on number of labeled trees with n vertices is known as **Cayley's theorem** after Arthur Cayley, British erudition who proposed it.

Theorem 17.5.3. *Number of labeled tree with n vertices is* n^{n-2}.

Proof. We present a bijective proof, which establishes a bijection between set of trees with vertex set S={1,2,....n} and the set S^{n-2}.

Here, S^{n-2} is the set of sequences of length *n-2* taking entries from the set S. Since $|S| = n$, there are exactly n^{n-2} ways to form sequences of length $n-2$ which take values from the set S. Hence, $|S^{n-2}| = n^{n-2}$.

Part 1: One-one map: $\underline{Trees \rightarrow S^{n-2}}$

The map is defined by algorithm commonly known as *Prufer's code.* Given a tree T we need to find $f(T) = (a_1, a_2, ...a_{n-2})$ where each $a_i \in S$.

At stage i we find the leaf with minimum label, add the neighbour of this leaf node to the sequence (as a_i) and delete that leaf node.

After $n-2$ iterations, only one of the $n-1$ edges remain and we have produced $f(T)$.

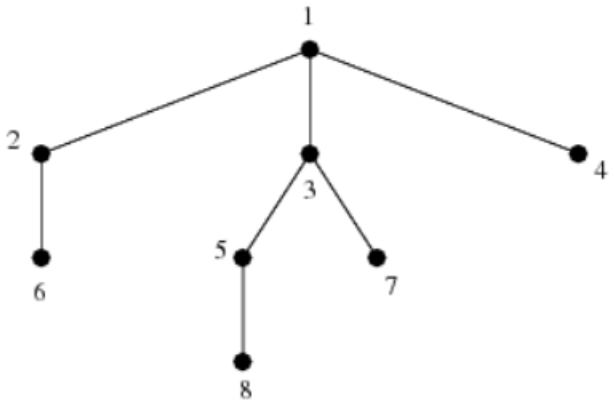

Figure 7: The Prufier code for this graph {1,2,1,3,3,5} which is formed by removing vertex 4,6,2,1,7 and 3 in that order.

Part 2: One-one map: $\underline{S^{n-2} \rightarrow Trees}$

Consider a sequence $s \in S^{n-2}$,$s = (a_1, a_2, ...a_{n-2})$ where each $a_i \in S$. Let m be minimum of the remaining elements not covered by $a_i's$. Connect a_1 to the m. Replace a_1 with m in the sequence.

We search for minimum element not in present updated sequence; connect a_2 with it; replace a_2 with it and thus update the sequence yet again.

Connect the last two remaining vertices not in the sequence to complete the tree.

We repeat the produce up to $n-2$ times, each time update the sequence.

Example: Consider the sequence s={1,2,1,3,3,5}; and the set of vertices S=1,2,..8.

Step 1: Minimum element of S not in s is 4. Connect 1-4. Updated sequence s={4,2,1,3,3,5}
Step 2: Minimum element of S not in s is 6. Connect 2-6. Updated sequence s={4,6,1,3,3,5}
Step 3: Minimum element of S not in s is 2. Connect 1-2. Updated sequence s={4,6,2,3,3,5}

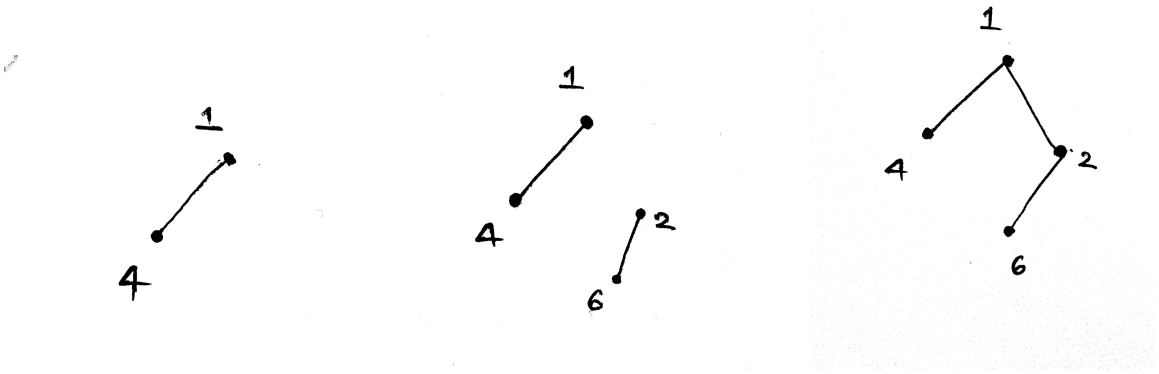

Figure 8: Steps 1-3

Step 4: Minimum element of S not in s is 1. Connect 1-3. Updated sequence s={4,6,2,1,3,5}
Step 5: Minimum element of S not in s is 7. Connect 3-7. Updated sequence s={4,6,2,1,7,5}
Step 6: Minimum element of S not in s is 3. Connect 5-3. Updated sequence s={4,6,2,1,7,3}.

Step 7: Since {5,8} are not present in the list we connect them to get the desired sequence.

The final graph is as in Figure 7.

Having explicitly defined the bijective map and its inverse we now it is evident that the number of labeled trees is same as $|S^{n-2}|$ which is

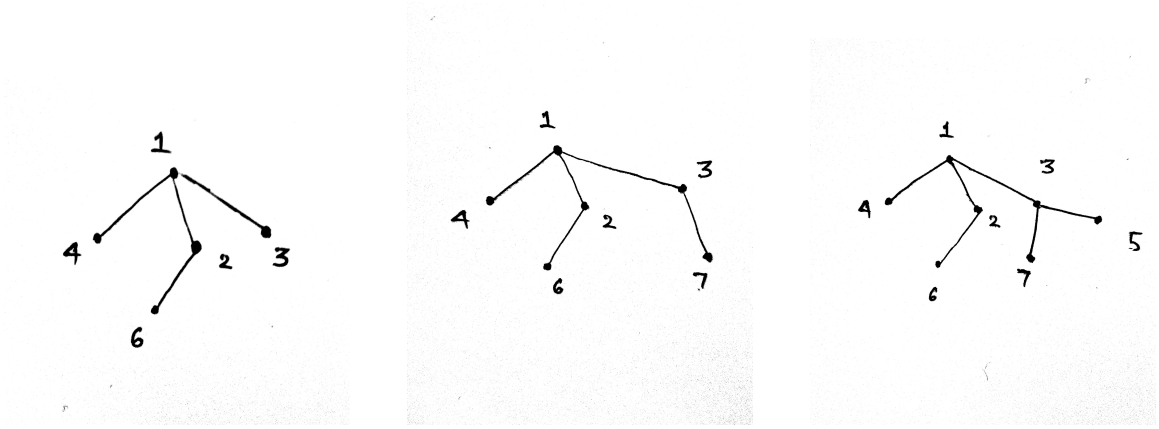

Figure 9: Steps 4-6

n^{n-2} thus completing the proof.

□

Theorem 17.5.4. *Number of labeled trees with n vertices and degree sequence $d_1, d_2, ...d_n$ is given by*

$$\frac{(n-2)!}{\prod_{i=1}^{n}(d_i-1)!}$$

Proof. While constructing the Prufier code of tree T we record x each time we delete the neighbour of x till we delete x itself or leave x among the last two vertices. Thus each vertex of x appears in $d_x - 1$ times in Prufier code.

Therefore we count trees with these vertex degrees by counting lists of length $n-2$ that for each i have d_i-1 copies of i. If we assign subscript to copies for each i to distinguish them, then we are permuting $n-2$ distinct objects and there are $(n-2)!$ list. Since copies of i are not distinguishable, we have counted each desired arrangement $\Pi_{i=1}^{n}(d_i-1)!$ many times, once in each way to order the subscript of each type.

□