Lecture 10
MATH-42021/52021 Graph Theory and Combinatorics.

Artem Zvavitch

Department of Mathematical Sciences, Kent State University

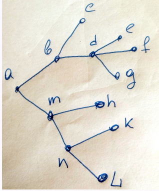
July, 2016.

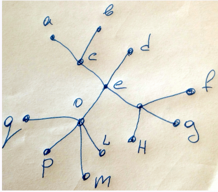
Trees - basic definitions

One of the most popular and useful special type of graphs is a tree

One of the most popular and useful special type of graphs is a tree

In undirected graphs, a *tree* is a connected graph with no circuits.

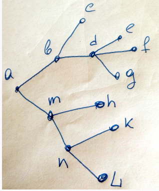


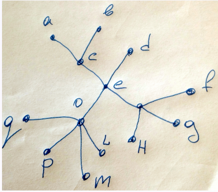


Trees - basic definitions

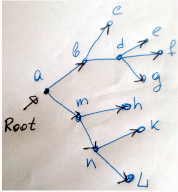
One of the most popular and useful special type of graphs is a tree

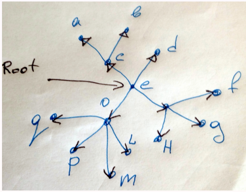
In undirected graphs, a *tree* is a connected graph with no circuits.



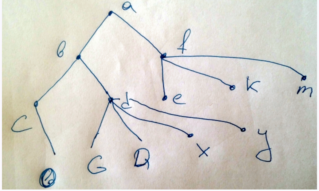


Another way to define a tree as a graph with special designated vertex - *root* such that there is a unique pass from a root to any other vertex in the tree. One can also use this definition for directed graphs.



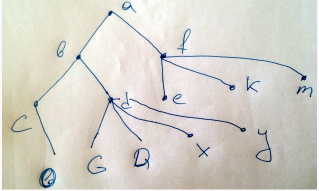


A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



Trees - basic definitions

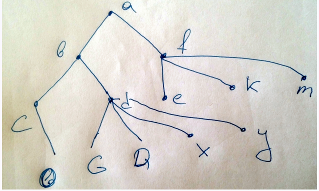
A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



then the vertices adjacent to a are placed one level below a (b and f in our example).

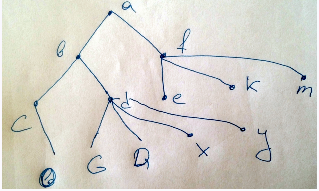
Trees - basic definitions

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



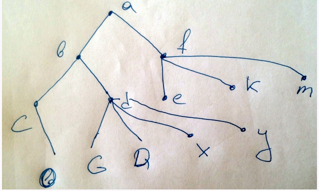
then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0.

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1.

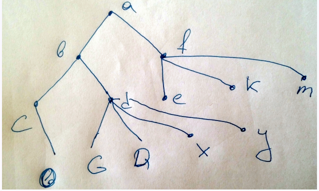
A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1. Vertices adjacent to level 1 (but not at level 0) are at level 2 (c, d, e, k, m on our example) and so forth.

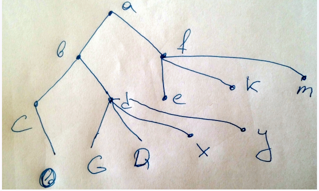
Trees - basic definitions

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



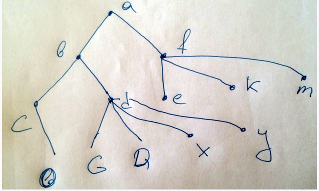
then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1. Vertices adjacent to level 1 (but not at level 0) are at level 2 (c, d, e, k, m on our example) and so forth. Note that the level number of vertex x in T is the length of the pass from root a to x.

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



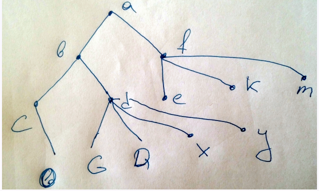
then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1. Vertices adjacent to level 1 (but not at level 0) are at level 2 (c, d, e, k, m on our example) and so forth. Note that the level number of vertex x in T is the length of the pass from root a to x.
For any vertex x, which is not the root, we say y is a parent of x if x and y are adjacent and level of y is one less then the level of x.

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1. Vertices adjacent to level 1 (but not at level 0) are at level 2 (c, d, e, k, m on our example) and so forth. Note that the level number of vertex x in T is the length of the pass from root a to x.
For any vertex x, which is not the root, we say y is a parent of x if x and y are adjacent and level of y is one less then the level of x. We also say, in this case, that x is a child of y. For example f is a parent of e and e, k, m are children of f.

A standard way to draw a rooted tree T is to place the root a at the top of the figure (see the picture below)



then the vertices adjacent to a are placed one level below a (b and f in our example). We say that root is at level 0. We also say that vertices adjacent to the root (b and f in our example) are at level 1. Vertices adjacent to level 1 (but not at level 0) are at level 2 (c, d, e, k, m on our example) and so forth. Note that the level number of vertex x in T is the length of the pass from root a to x.
For any vertex x, which is not the root, we say y is a parent of x if x and y are adjacent and level of y is one less then the level of x. We also say, in this case, that x is a child of y. For example f is a parent of e and e, k, m are children of f.
Vertices of T with no children are called *leaves* of T (O, G, Q, x, y in our example).
Vertices with children are called *internal* vertices.

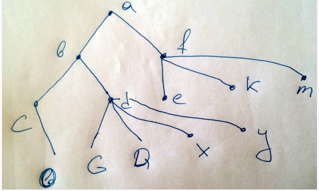
Theorem.

A tree with n vertices has $n-1$ edges.

Theorem.

A tree with n vertices has $n-1$ edges.

Proof : We may assume that the tree is rooted (we select the root and structure the tree as before).



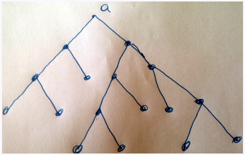
Then we can pair every vertex (but the root!) with unique incoming edge from its parent. This gives exactly $n-1$ edges.

□

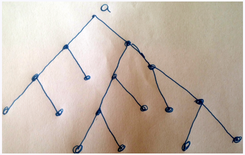
Trees - some basic theorems

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*.

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



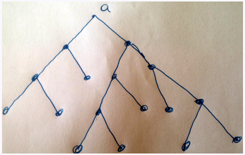
If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.

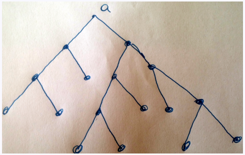


Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent).

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.

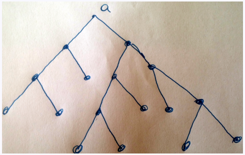


Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



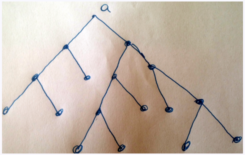
Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

□

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

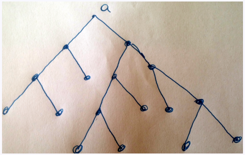
Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

□

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

□

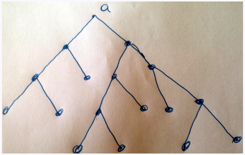
Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.

Trees - some basic theorems

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

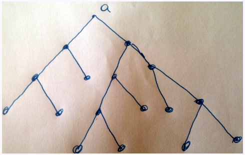
□

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

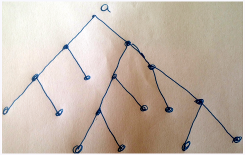
□

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n + 1]m$.

If each internal vertex of a rooted tree has m children, we call T an *m-ary tree*. If $m = 2$, then T is a *binary tree*.



Theorem.

Let T be an m-ary tree with n vertices, of which i vertices are internal. Then $n = mi + 1$

Proof : Each vertex in the tree is a child of unique vertex (its parent). Note that each of the internal vertices has m children, so there are in total $m \times i$ children . Adding the one childless vertex (root) we finish the proof of the theorem.

□

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n + 1]m$.

The proof follows immediately from the theorem and the fact that $l + i = n$.

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i+1$ and $n = mi+1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n+1]m$.

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

The tournament schedule and plan can be managed as a binary-tree.

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

The tournament schedule and plan can be managed as a binary-tree. The entrants are leaves and the matches are internal vertices, the Winner = root.

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l = (m-1)i + 1$ and $n = mi + 1$.
- Given l, then $i = (l-1)/(m-1)$ and $n = (ml-1)/(m-1)$.
- Given n, then $i = (n-1)/m$ and $l = [(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

The tournament schedule and plan can be managed as a binary-tree. The entrants are leaves and the matches are internal vertices, the Winner = root. So our goal is to find i,

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l=(m-1)i+1$ and $n=mi+1$.
- Given l, then $i=(l-1)/(m-1)$ and $n=(ml-1)/(m-1)$.
- Given n, then $i=(n-1)/m$ and $l=[(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

The tournament schedule and plan can be managed as a binary-tree. The entrants are leaves and the matches are internal vertices, the Winner = root. So our goal is to find i, with $m=2$ and $l=48$.

Corollary

Let T be an m-ary tree with n vertices, of which i vertices are internal and l are leaves. Then knowing just one of those parameters you may compute other two:

- Given i, then $l=(m-1)i+1$ and $n=mi+1$.
- Given l, then $i=(l-1)/(m-1)$ and $n=(ml-1)/(m-1)$.
- Given n, then $i=(n-1)/m$ and $l=[(m-1)n+1]m$.

Example

Assume 48 people signed up for chess tournament, how many matches will be played in the tournament (the rules that once you loose you are "out" of the game).

The tournament schedule and plan can be managed as a binary-tree. The entrants are leaves and the matches are internal vertices, the Winner = root. So our goal is to find i, with $m=2$ and $l=48$. Using corollary we get
$i=(l-1)/(m-1)=47/1=47$.

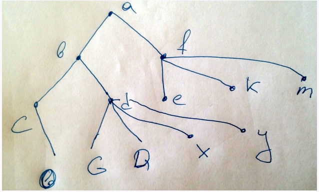
More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$:

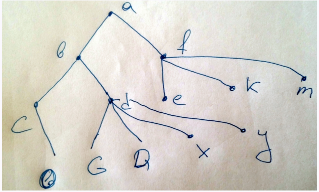
The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$:
This one balanced:



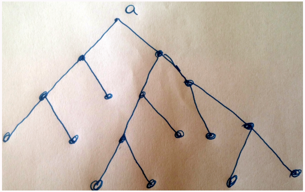
More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$:

This one balanced:



This one NOT



More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \le m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \ge \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h.

More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.
Now assume that the statement is true for an m-ary try of hight h,

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.

Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.
Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.
Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents. Indeed if we remove the leaves of T we get an m-ary tree T' of hight h, and can apply an induction to get that the number of leaves of this tree is at most m^h.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.

Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents. Indeed if we remove the leaves of T we get an m-ary tree T' of hight h, and can apply an induction to get that the number of leaves of this tree is at most m^h. Now we note that each leave of T' may have at most m children as a parent in T.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.
Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents. Indeed if we remove the leaves of T we get an m-ary tree T' of hight h, and can apply an induction to get that the number of leaves of this tree is at most m^h. Now we note that each leave of T' may have at most m children as a parent in T. So the number of leaves in T is at most $m^h \times m = m^{h+1}$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.

Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents. Indeed if we remove the leaves of T we get an m-ary tree T' of hight h, and can apply an induction to get that the number of leaves of this tree is at most m^h. Now we note that each leave of T' may have at most m children as a parent in T. So the number of leaves in T is at most $m^h \times m = m^{h+1}$.

Finally, if all of the leaves of T are at hight $h+1$, then all of the leaves of T' are at hight h so, applying induction,

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$.

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : To prove the first statement we will use the induction on h. The statement is obvious if $h = 0$, and also not so hard if $h = 1$, indeed in this case the tree will have one root (as always) and the root will have m leaves, thus $l = m^1$.

Now assume that the statement is true for an m-ary try of hight h, our goal is to prove it for an m-ary tree T of hight $h+1$. Consider all leaves of T, note that all of them together may have at most m^h parents. Indeed if we remove the leaves of T we get an m-ary tree T' of hight h, and can apply an induction to get that the number of leaves of this tree is at most m^h. Now we note that each leave of T' may have at most m children as a parent in T. So the number of leaves in T is at most $m^h \times m = m^{h+1}$.

Finally, if all of the leaves of T are at hight $h+1$, then all of the leaves of T' are at hight h so, applying induction, we get that the number of leaves of T' is exactly m^h and thus the number of leaves of T is exactly m^{h+1}.

More basic theorems

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h,

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$,

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$, but h is an integer number so $\lceil \log_m l \rceil \leq h$.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$, but h is an integer number so $\lceil \log_m l \rceil \leq h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$, but h is an integer number so $\lceil \log_m l \rceil \leq h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$ and the number of leaves in T' is exactly m^{h-1}.

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$, but h is an integer number so $\lceil \log_m l \rceil \leq h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$ and the number of leaves in T' is exactly m^{h-1}. But, then the number of leaves in T is at least $m^{h-1} + (m-1)$ (T is oh hight h so must have some leaves at hight h so at least one leave of T' must have children).

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \leq m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \geq \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \leq h$, but h is an integer number so $\lceil \log_m l \rceil \leq h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$ and the number of leaves in T' is exactly m^{h-1}. But, then the number of leaves in T is at least $m^{h-1} + (m-1)$ (T is oh hight h so must have some leaves at hight h so at least one leave of T' must have children). Thus

$$m^{h-1} + (m-1) \leq l \leq m^h$$

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \le m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \ge \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \le h$, but h is an integer number so $\lceil \log_m l \rceil \le h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$ and the number of leaves in T' is exactly m^{h-1}. But, then the number of leaves in T is at least $m^{h-1} + (m-1)$ (T is oh hight h so must have some leaves at hight h so at least one leave of T' must have children). Thus

$$m^{h-1} + (m-1) \le l \le m^h$$

or

$$\log_m(m^{h-1} + (m-1)) \le \log_m l \le h,$$

The *hight* of a rooted tree is the length of the longest path from the root. Equally we can see it as the largest existing level number. A tree, of hight h, is called balanced it all existing leaves are at level h or $h-1$. We also will denote by $\lceil r \rceil$ the smallest integer which is greater or equal to r. For example $\lceil 4.7 \rceil = 5$ and $\lceil 10 \rceil = 10$

Theorem.

Let T be an m-ary tree of hight h with l leaves. Then

- $l \le m^h$, and if all leaves are at height h, then $l = m^h$.
- $h \ge \lceil \log_m l \rceil$, and if the tree is balanced, $h = \lceil \log_m l \rceil$.

Proof : The second statement of the Theorem follows from the first one. Indeed, we notice that the number of leaves is less or equal to m^h, thus $\log_m l \le h$, but h is an integer number so $\lceil \log_m l \rceil \le h$. Now if the tree T is balanced it means that if we remove all leaves at level h we will get a tree T' such that all leaves of T' are at level $h-1$ and the number of leaves in T' is exactly m^{h-1}. But, then the number of leaves in T is at least $m^{h-1} + (m-1)$ (T is oh hight h so must have some leaves at hight h so at least one leave of T' must have children). Thus

$$m^{h-1} + (m-1) \le l \le m^h$$

or

$$\log_m(m^{h-1} + (m-1)) \le \log_m l \le h,$$

but $\log_m(m^{h-1} + (m-1)) > \log_m(m^{h-1}) = h-1$, so $\log_m l \in (h-1, h]$, thus $h = \lceil \log_m l \rceil$.

□

An example

Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even).

An example

Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even). We would like for given n to provide a fastest (minimal number of weightings) algorithm to find the counterfeit coin.

An example

Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even). We would like for given n to provide a fastest (minimal number of weightings) algorithm to find the counterfeit coin. To make a problem a bit easier assume that we do know that a coin is too light.

An example

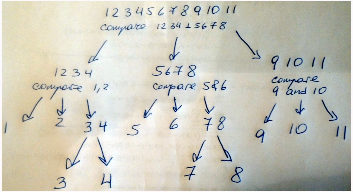
Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even). We would like for given n to provide a fastest (minimal number of weightings) algorithm to find the counterfeit coin. To make a problem a bit easier assume that we do know that a coin is too light.

The idea is to form a tree. The approach that we can always divide the coins into three almost equal piles (for example round $n/3$ and make one pile to contain the reminder). Compare two piles with the same number of coins. If the weight equal - bad coin is in the third pile. If one is lighter there your bad coin.

An example

Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even). We would like for given n to provide a fastest (minimal number of weightings) algorithm to find the counterfeit coin. To make a problem a bit easier assume that we do know that a coin is too light.

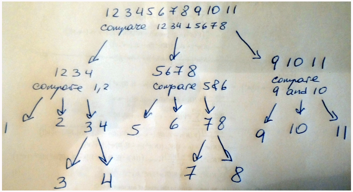
The idea is to form a tree. The approach that we can always divide the coins into three almost equal piles (for example round $n/3$ and make one pile to contain the reminder). Compare two piles with the same number of coins. If the weight equal - bad coin is in the third pile. If one is lighter there your bad coin.



An example

Assume we have n coins, one of which is counterfeit, too light or too heavy, and a balance to compare the weight of any two sets of coins (the balance can tip to the right, to the left or to be even). We would like for given n to provide a fastest (minimal number of weightings) algorithm to find the counterfeit coin. To make a problem a bit easier assume that we do know that a coin is too light.

The idea is to form a tree. The approach that we can always divide the coins into three almost equal piles (for example round $n/3$ and make one pile to contain the reminder). Compare two piles with the same number of coins. If the weight equal - bad coin is in the third pile. If one is lighter there your bad coin.



This way we create a ternary tree with n leaves, thus the hight must be $\lceil \log_3 n \rceil$.

How many (undirected) trees are there?

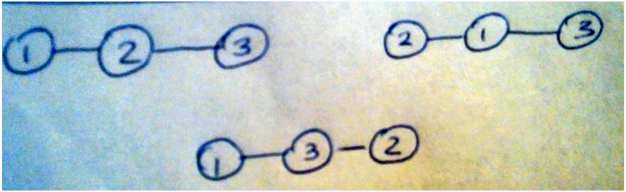
Consider n items (we will denote them simply by numbers $1, 2, \ldots, n$).

Consider n items (we will denote them simply by numbers $1, 2, \ldots, n$). So how many different, undirected trees can you build with those items as the vertices?

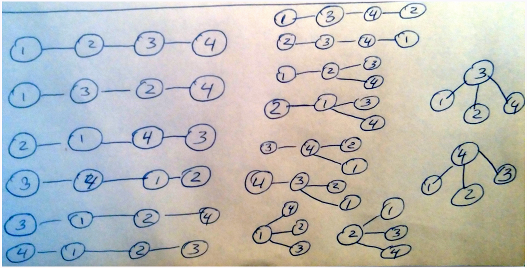
How many (undirected) trees are there?

Consider n items (we will denote them simply by numbers $1, 2, \dots, n$). So how many different, undirected trees can you build with those items as the vertices? The case $n = 2$ is trivial (just one tree, build by connecting two vertices with one edge).

Consider n items (we will denote them simply by numbers $1, 2, \ldots, n$). So how many different, undirected trees can you build with those items as the vertices? The case $n = 2$ is trivial (just one tree, build by connecting two vertices with one edge). What about $n = 3$:



Consider n items (we will denote them simply by numbers $1, 2, \ldots, n$). So how many different, undirected trees you can build with those items as the vertices? The case $n = 2$ is trivial (just one tree, build by connecting two vertices with one edge). What about $n = 4$:



How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n

How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).

How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows.

How many (undirected) trees are there – general theorem.

Theorem

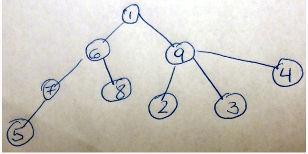
There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.

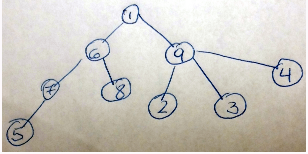


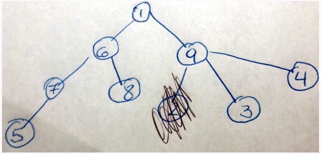
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



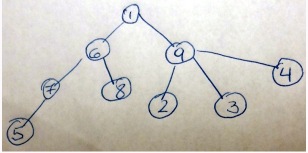


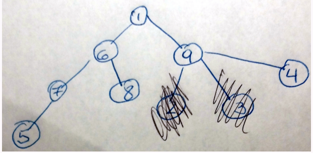
Now we remove the vertex l_1 and repeat the process.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



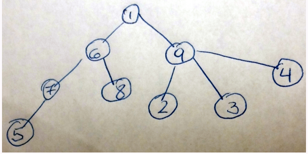


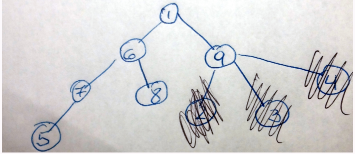
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$;

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



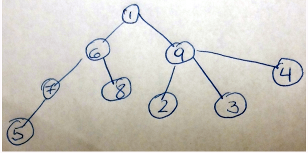


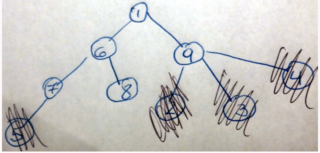
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$;

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



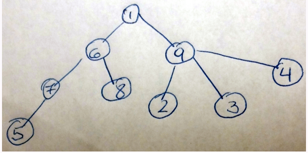


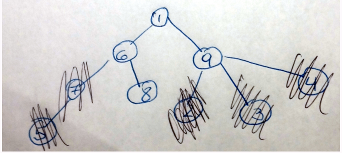
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$;

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



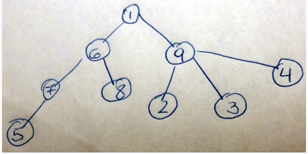


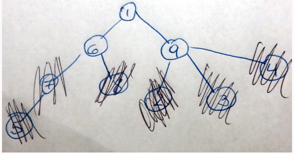
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$;

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



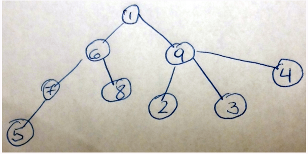


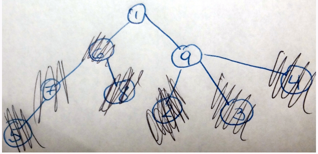
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$;

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





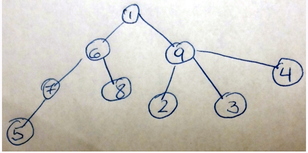
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$.

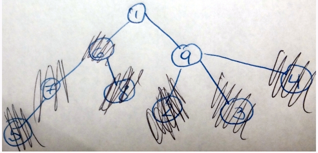
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



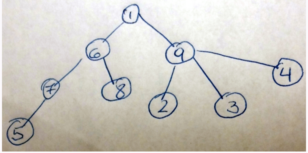


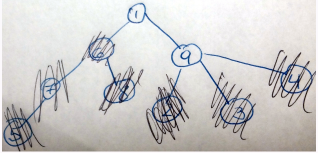
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.



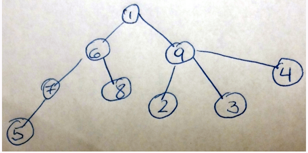


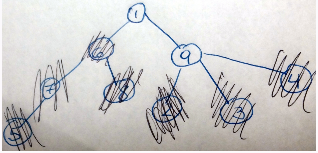
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences*.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \ldots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





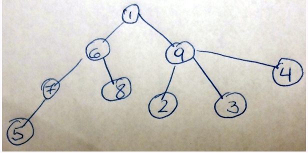
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences.*

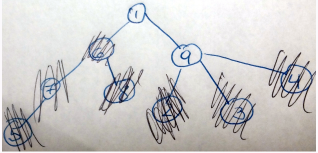
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





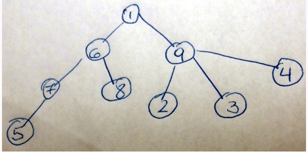
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences*.
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process.

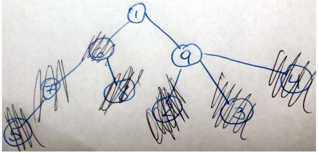
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





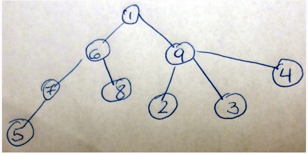
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences*.
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process. Observe that leaves (vertices of degree 1) will never appear in the sequence.

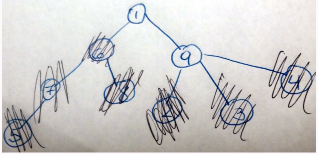
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





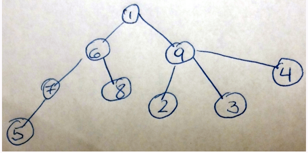
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences.*
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process. Observe that leaves (vertices of degree 1) will never appear in the sequence. So we draw them first.

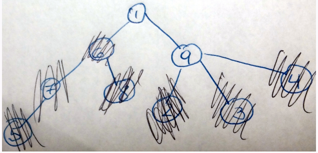
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





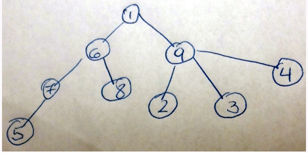
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences.*
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process. Observe that leaves (vertices of degree 1) will never appear in the sequence. So we draw them first. Next observe that the first number of the sequence (9 in our case) is the neighbor of the smallest numbered leaf (leaf with number 2 in our example)

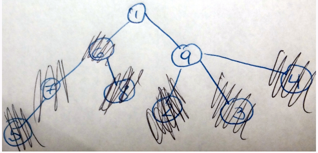
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





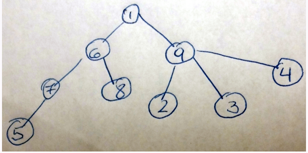
Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences.*
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process. Observe that leaves (vertices of degree 1) will never appear in the sequence. So we draw them first. Next observe that the first number of the sequence (9 in our case) is the neighbor of the smallest numbered leaf (leaf with number 2 in our example)

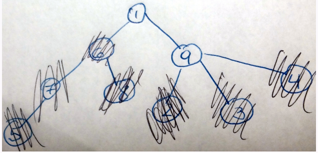
How many (undirected) trees are there – general theorem.

Theorem

There are n^{n-2} different undirected trees on n items.

Proof : First notice that n^{n-2} is a number of different sequences of length $(n-2)$ where each element is selected out of n (notice that we do not ask elements of the sequence to be different, thus we can select the first element in n ways, after the second element in n ways and continue till $(n-2)$'nd element, we also note that the order does meter for sequences).
Also note, that we now can prove the theorem by creating a one to one correspondence between trees of n-elements and sequences of n-elements. For any tree of n elements we will create a sequence $(s_1, \dots, s_{n-2})$ of length $n-2$ as follows. Let l_1 be a leaf of the tree with smallest number and let s_1 be the number of the one vertex adjacent to it. For example on the tree below l_1 is a leaf with number 2 on it and $s_1 = 9$.





Now we remove the vertex l_1 and repeat the process. For our example we will get $l_2 = 3$ and $s_2 = 9$; $l_3 = 4$ and $s_3 = 9$; $l_4 = 5$ and $s_4 = 7$; $l_5 = 7$ and $s_5 = 6$; $l_6 = 8$ and $s_6 = 6$; $l_7 = 9$ and $s_7 = 1$. We got sequence $(9, 9, 9, 7, 6, 6, 1)$ corresponding to our graph. Such sequence are called *Prufer Sequences.*
Our next goal is to show that any such $(n-2)$-length sequence defines a unique n-item tree. To do this we simply reverse the process. Observe that leaves (vertices of degree 1) will never appear in the sequence. So we draw them first. Next observe that the first number of the sequence (9 in our case) is the neighbor of the smallest numbered leaf (leaf with number 2 in our example) we