

Advanced Graph Theory
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture - 06
Spanning Tree and Enumeration

(Refer Slide Time: 00:18)

Preface

Recap of previous Lecture:

- In the previous lecture, we have discussed the basic properties of trees and distance.

Content of this Lecture:

- In this lecture, we will discuss the Prüfer code, Cayley's formula, counting of spanning trees using various methods including Matrix Tree Theorem.

Advanced Graph Theory Spanning Trees and Enumeration

Lecture 6 spanning trees and enumeration. Recap of previous lecture previous lecture, we have discussed the basic properties of trees and distance content of this lecture in this lecture we will discuss the Prüfer code, Cayley's theorem counting of spanning trees using various methods including matrix tree theorem.

(Refer Slide Time: 00:42)

Spanning Trees and Enumeration 2.2

- There are $2^{\binom{n}{2}}$ simple graphs with vertex set $[n]=\{1,\dots,n\}$, since each pair may or may not form an edge.

- How many of these are trees?

- With vertex set $[n]$, there are n^{n-2} labelled trees; this is **Cayley's Formula.**

$2^{\binom{n}{2}}$ - Simple Graphs

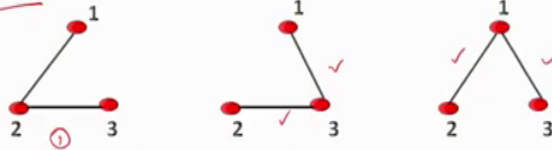
(i) Total Simple $2^{\binom{n}{2}}$
 (ii) Trees $[n]$ n^{n-2}
 Cayley's Formula.

Spanning trees and enumeration there are $2^{\binom{n}{2}}$ simple graphs with vertex set, which are labeled from 1 to n since, each pair may or may not form an edge therefore, total number of simple graphs which are required or which will be formed is $2^{\binom{n}{2}}$. Now, out of this simple graphs how many are basically the trees? How many are the trees? So, that is given by Cayley's formula. So, with the set vertices, which are labeled as from 1 to n, that is denoted by this symbol. So, then they are n^{n-2} different labeled trees and that is called the Cayley's formula. So, the total number of possible simple graphs of n vertices is $2^{\binom{n}{2}}$ and out of these graphs how many are basically different labeled trees? Which are labeled the vertices from 1 to n they are n^{n-2} different trees are there, that is given by Cayley's formula.

(Refer Slide Time: 02:25)

Enumeration of Trees 2.2

Example: $n = 3$



- With one or two vertices, only one tree can be formed.
- With three vertices there is still only one isomorphism class, but there are three trees with vertex set [3]. $n^{n-2} = 3^1 = 3$
- With vertex set [4], there are four stars and 12 paths, yielding 16 trees. $\checkmark \rightarrow$
- With vertex set [5], a careful study yields 125 trees.

Now, using Cayley's formula we can see that, if there are 3 different vertices and we label them as 1 2 3. So, how many different trees will be possible? So, 3 raise power 1, that is 3 different trees are possible. So, this is 1 possible tree then, these the node 1 and 2 and 2 and 3 they have got an edge in another possibility if 1 and 3 and 2 and 3 they have got an edge, that possibility when 1 on 2 and 1 and 3 have got an edge. So, there are 3 possibilities only with the Labeled trees. Similarly, if we take a vertex set 4 vertices and they are Labeled out of this set 4 then, there are 4 stars and 12 different paths in turn they comes out to be 16 different trees as per as Cayley's formula is concerned and so on.

(Refer Slide Time: 03:31)

Cayley's Formula and Prüfer Code

- Now we may see a pattern. With vertex set $[n]$, there are n^{n-2} trees; this is **Cayley's Formula**. Prüfer, Kirchhoff, Pólya, Renyi, and others found proofs.
- We will discuss a bijective proof, establishing a one-to-one correspondence between the **set of trees with vertex set $[n]$** and **a set of known size**.
- Given a set S on n numbers, there are exactly n^{n-2} ways to form a list of length $n-2$ with entries in S . The set of lists is denoted S^{n-2} .
- We use S^{n-2} to encode the trees with vertex set S . The list that results from a tree is its **Prüfer code**.

So now we can see a pattern with the given vertex set n , and using this particular formula that particular pattern is basically covered, that is n raise to the power n minus 2 and that is called Cayley's formula. Prufer, Kirchhoff, Polya and Renyi they have formed out the proofs we are going to discuss all this in this part of this lecture, we will discuss bijective proof establishing one to one correspondence between the set of trees with the vertex set n and another set of known size. So, given a set n set on n numbers there are exactly n raise power n minus 2 different ways to form a list of length n minus 2 with the entries in S . So, the set of list is denoted by S raise power n minus 2 we use S raise power n minus 2 to encode the trees with the vertex set S the list, that results from the tree is it is Prufer code.

(Refer Slide Time: 04:43)

Algorithm 2.2.1. (Prufer code) Production of $f(T) = (a_1, \dots, a_{n-2})$

Algorithm. (Prufer code) Production of $f(T) = (a_1, \dots, a_{n-2})$

Input: A tree T with vertex set $S \subseteq [N]$

Iteration: At the i th step, delete the least remaining leaf, and let a_i be the neighbor of this leaf.

1. Delete 2 $a_1 = 7$
2. Delete 3 $a_2 = 4$
3. Delete 5 $a_3 = 4$
4. Delete 4 $a_4 = 1$
5. Delete 6 $a_5 = 7$
6. Delete 7 $a_6 = 1$

Advanced Graph Theory Spanning Trees and Enumeration

Let us see the algorithm for Prufer code. So, the Prufer code will produce code of length n minus 2, that is these numbers a_1, a_2 and so on up to a_{n-2} these numbers are drawn out of this particular set n and so given a tree this particular function f will produce a code and that is called a Prufer code. So, for a given tree there is code which will be of length n minus 2 that is called a Prufer code. So, the input to this algorithm is a tree with the vertex set S , which is a subset of this particular label, and which is given as per as this algorithm is concerned at i th step it will delete the last remaining leaf and a_i is basically the neighbor of this particular leaf, that will be noted down. This particular iteration keeps on iterating finally, it will terminate when there will be only one edge

joining between 2 vertices remains or you can also say up to K_2 graphs will remain then only the algorithm terminates with a successful Prüfer code of Labeled tree.

(Refer Slide Time: 06:12)

Generation of Prüfer Sequence from Labelled Tree

Proof Idea: Labelled tree $T \leftrightarrow$ Prüfer sequence S
 1:1 correspondence on n elements of length $n-2$

• How many such sequence? $n \times n \times n \times \dots \times n = n^{n-2}$

Algorithm: Labelled tree $T \rightarrow$ Prüfer Sequence S

- Find a leaf of T with smallest label \rightarrow (about 2 leaves)
- \rightarrow add the neighbour to sequence S
- \rightarrow delete this leaf
- Repeat until our tree is K_2

Handwritten notes: $[n] = \{1, n\}$, $n \times n \times n \times \dots \times n = n^{n-2}$, $n-2$, n .

Advanced Graph Theory | Spanning Trees and Enumeration

So, let us see the different aspects involved in this Prüfer code algorithm. So, first we will see one aspect that is, how to generate a Prüfer sequence from a given Labeled tree? So, the so; that means, we are given a Labeled tree and we have to generate a Prüfer sequence S and on the other side also if a Prüfer sequence S is given we can generate that, unique tree which is Labeled tree. So, there is one to one correspondence from tree to a particular Prüfer sequence and Prüfer sequence to a tree we can construct. Now, if that is the case for a given tree, we are going to produce a Prüfer sequence and these sequence will be drawn from each element of this Prüfer sequence will be from 1 to n and total length of this particular code S will be n minus 2. So, how many such sequences are possible? So, let us see that, if the length of this particular sequence is n minus 2; that means, there are n minus 2 different containers and each element of this container you can fill it, with the number which are ranging from 1 to n , that is the label of these vertices. Similarly, in the other container also we can replace or we can choose any of these n numbers and so on so; that means, n multiplied by n and so on up to n minus n times if you multiply that becomes, n raise power n minus 2. So, how many such sequences are possible? They are basically in this number, that is n raise power n minus 2 at every sequence represents a specific Labeled trees. So, how many trees will be there that is these number of spanning trees are possible.

Now, we have to see this particular algorithm, that given Labeled tree how we are going to generate the Prufer sequence? So, the algorithm the first step of the algorithm is find a leaf of a tree with the smallest label this particular; that means, whenever a tree is given always there are 2 leafs that, we have seen in the previous theorem 2 leafs whenever there is a in the tree. So, this particular assumption to find out a leaf is possible whenever a tree is given.

So, find a leaf of T with the smallest label. So, having identified a smallest label leaf what we will do? We will delete this particular leaf and the neighbor of this particular leaf will be added to that particular sequence. So, we are building up that sequence this particular step is repeated we have to keep on repeating iterate till this condition arises that, our tree becomes only K_2 ; that means, only one edge connecting to different vertices basically is left out at, that time this particular algorithm will terminate.

(Refer Slide Time: 10:00)

Example

$T = T(\{2, 3, 4, 5\})$

K_2

$1 \quad 7 \quad 3-1$

S = ()

S = (1)

S = (1, 7)

S = (1, 7, 6)

S = (1, 7, 6, 6)

S = (1, 7, 6, 6, 1) ←

Why n-2? ✓

S = (1, 7, 6, 6, 1) ← Terminate

$[1] = 5$

$= n-2$

2 times
ie $d(6) - 1 = 2$
 $d(1) - 1 = 0$

Advanced Graph Theory
Spanning Trees and Enumeration

Let us see this particular example, let us say this T is given to us and we want to generate a Prufer sequence. So, first we have to see in this particular tree how many different we have to see this tree? How many different leafs are there? This is one leaf another leaf. So, there are 4 set of leafs and out of this 4 this is one is the leaf with a least value 2. So, what we are going to do? We will take this particular neighbor, that is this particular neighbor we are going to take it and this will be removed and the remaining tree, that is

let us say after one iteration that becomes, T prime T minus the vertex, which is label to will be deleted.

So, this particular tree T prime will have how many other leafs? 3 4 5 there are 3 different leafs, which are remaining and the one with a least value is 3. So, 3 will be removed, but the neighbor of 3 is 7, 7 will be added in the sequence S . So, 3 also will be removed from this particular tree. So, 2 and 3 they will be removed. So, the in the resulting tree there are how many leafs are there? 1 2 3 different leafs are there and out of them the least value is 4. So, the neighbor of 4 is 6. So, 6 will be included and now, 4 will be removed from the tree and the resulting tree will basically get 4 removed out of it.

Now, the remaining leaf nodes will be 7 and 5 out of them 5 is having the least value. So, the neighbor of 5 is 6. So, 6 will be added again. So, you see some of the vertices, some of the labels are repeated in the sequence S or the Prufer code and then we will remove it from. So, the resulting tree will be the T minus 2 3 4 and 5. So, the leafs which will be remaining is 6 and 7 in that resulting tree. So, out of 6 and 7 the least value is 6. So, it will be picked up and the neighbor of 6 is 1.

So, 1 will be put and 6 will be deleted. So, after that deletion there will be only one edge remains, which is connecting 1 and 7, and that is nothing but $2K2$. So, this particular algorithm will terminate at this end terminate with a sequence S . So, if we count how many elements are there in the particular sequence? 1, 2, 3, 4, 5 and how many nodes were there? They were 7. So, that 5 is nothing but n minus 2. So, the length of the Prufer sequence as we have seen in the algorithm that, it will generate for a particular tree a Prufer sequence of length Prufer sequence S of length n minus 2. Let us see, why this n minus 2 is basically coming up as the Prufer sequence length.

(Refer Slide Time: 14:08)

Contd...

OBSERVATIONS

- No leaf gets appended to S ✓
- Every vertex v is added to S a total of $\deg(v)-1$ times ✓

So in given Tree let n vertices, m edges $\Rightarrow m = n - 1$ ✓

Number of terms in S - length ✓

$$\begin{aligned}\sum_{v \in v(T)} (\deg(v) - 1) &= \left(\sum_{v \in v(T)} \deg(v) \right) - \sum_{v \in v(T)} (1) \\ &= 2m - n \\ &= 2(n-1) - n \\ &= n - 2\end{aligned}$$

Proved

So, in the observation we have seen there is no leaf node, which is appended to that particular sequence S and also, we have seen another observation that, every vertex v is added to S a total of how many times? Degree of that vertex minus 1 times. So, we can see. So, 6 is appearing twice, why? Because, the degree of 6 node number 6 is having 3. So, 3 minus 1 that is 2 times, that is the degree of this node 6 minus 1 that is 2 times it is repeating. Similarly, here, the node number 1 is also repeated 2 times because, of the same condition that the degree of is equal to degree of 1 minus 1 becomes 2. So, 2 times it is appearing.

So, with these 2 observations what we can see that, for a given tree T having n different vertices or the labels and m different edges since, it is a tree we know that this m is equal to n minus 1. Now, let us see how many terms are there in S ? So, number of terms which will appear in the Prufer sequence S is nothing but, degree minus 1 of that vertex and for all the vertices we are going to make a sum, that will adapt to the length of the Prufer sequence. So, with this particular summation if we sum the degrees for all the vertices this is nothing but, a hand shaking lemma this will result in twice the number of edges similarly, one if we add how many times if we add n times it will become n . So, $2m$ if we substitute the value of m is n minus 1. So, this will give you the length as n minus 2. So, the length of the Prufer code is n minus 2 that we have proved.

(Refer Slide Time: 16:44)

Generation of Labelled Tree from Prüfer Sequence

Proof Idea: Prüfer sequence S of length $n-2$ using symbols $1, \dots, n$
 \rightarrow Labelled tree T

- Let $S = (a_1, a_2, \dots, a_{n-2}), a_i \in \{1, \dots, n\}$

Algorithm: Prüfer Sequence $S \rightarrow$ Labelled tree T

- Find smallest element $x \in \{1, \dots, n\} \setminus S$
 \rightarrow join x to 1st element of S
 \rightarrow delete a_1 from S , delete x from $\{1, \dots, n\}$
- Find smallest $y \in L \setminus \{x\}$ not in $S \setminus \{a_1\}$
 \rightarrow join y to the 1st element of $S \setminus \{a_1\}$
 \rightarrow delete a_2 from S
 \rightarrow delete y from L
- Continue until 2 items remain in L
 \rightarrow join these two with an edge

Handwritten notes:
 $L = \{1, \dots, n\}$
 $S = (a_1, a_2, \dots, a_{n-2})$
 $L = L \setminus \{x\}$
 $S = S \setminus \{a_1\}$

Advanced Graph Theory **Spanning Trees and Enumeration**

So, before we go ahead let us see that, if you are given Labeled if you are given a Prüfer sequence how we are going to generate a label tree from the Prüfer sequence? Let us see the proof idea, let us assume that the Prüfer sequence of n minus 2 length is given using this symbols 1 to n then we are going to generate a unique label tree T . Let us assume that, S contains a_1, a_2 and so on up to a n minus 2, why? Because, only the length of the code is n minus 2 and these values of a_i is drawn from 1 to n . Let us see the steps of this algorithm that once the Prüfer sequence is given how we are going to generate a label tree from a Prüfer sequence of length n minus 2 which is having the symbols from 1 to n .

So, the first step is to find out the smallest element x from this particular set of label that is 1 to n , which is not present in that particular Prüfer sequence S which is given to us. So, let us assume that, it is that particular label S which is not present in S . So, join this particular element to the first element of S and delete a_1 from S and delete x also from the set of labels. So, we will run 2 different 2 different sets one set let us call it as L , the other set is a Prüfer code that is a_1, a_2 and so on up to a n minus 2. So, we will take this particular value of x and we will delete this x . Similarly, here, the first element also will be deleted. So, the resulting list L of labels, and that the sequence S will be after removing the element a_1 .

So, now the next iteration will began from this particular modified list L prime and S prime again we are going to find out a smallest particular label here, which is not present

in the modified sequences prime and we are going to join this particular element, which is having the smallest label with the first element of S prime and we add an edge and then, we will remove these 2 elements from this particular list again we are modifying and we continue to iterate until 2 items remain in this particular L then, these 2 items when they will remain we are going to add an edge and we conclude generating all label tree out of a given Prufer code.

(Refer Slide Time: 20:36)

Example

$S = (1, 7, 6, 6, 1)$ length 5 $\rightarrow n = 7$
 $L = \{1, 2, 3, 4, 5, 6, 7\}$ vertices with labels
Handwritten: (1, 7, 6, 6, 1) are circled in green. Note: "least label not in S"

$S \rightarrow T$

$T \rightarrow S$

$x \in L \setminus \{S\}$ smallest $\rightarrow x = 2$
 (edge 12, remove 2 from L and 1 from S)
 $S = (7, 6, 6, 1)$
 $L = (1, 3, 4, 5, 6, 7)$

Handwritten:
 $Prufer \text{ code } S \Rightarrow T$
 $S \Rightarrow T$

Advanced Graph Theory
Spanning Trees and Enumeration

Let us take an example, if the Prufer code let us say 1 7 6 6 1 is given. Here, we can see that length is 5 and the label set is L 1, 2 and so on up to 7. So, the smallest label here is 2, which is not present in S. So, what we will do? We will take and we will take the first element of S 1 S and we will add n edge between 1 and 2 then, we will remove it and that resulting list S and L is go will be considered for the next iteration, in the next iteration what we will see that, 3 is the label which is not present in S. So, let us consider 3 and we are going to basically join it with 7.

So, 3 and 7 will contribute to add an edge and now, they will be removed from both L and S. Similarly, let us take the label number 4, 4 is not present in S. So, 4 is the least label, which we will consider and we will add an edge with 6. So, 6 and 4 will be placed and now, we are going to remove these 2 notes from this particular list. So, the remaining elements in list L, let us say, the label number 5 is not present in 6 and 1. So, 5 will be

chosen up and the first element in S is 6 both will be joining with an edge and now, they will be removed then we are going to consider the label number 6, 6 is not present in S .

So, 6 and 1; they will get an edge. So, 6 will be removed and 1 also will be removed. Now finally, we are left with 1 and 7. So, 1 and 7 will be placed an edge and we terminate. So, when we terminate we see that, we have generated out of S we have produced a tree. So, given a Prufer code S , we are we have produced a tree corresponding tree, that we have already seen in the in the previous proof that, if let us say a tree is given how we are going to produce a Prufer code and now, we have seen given a Prufer code we have to generate we can generate a unique tree out of that code, that becomes a Cayley's formula.

(Refer Slide Time: 23:55)

Theorem: Cayley's Formula [1889] 2.2.3

Theorem: For a set $S \subseteq N$ of size n , there are n^{n-2} trees with vertex set S

Proof: (Prüfer [1918]).

This holds for $n = 1$, so we assume $n \geq 2$. We prove that Algorithm 2.2.1 defines a bijection f from the set of trees with vertex set S to the set S^{n-2} of lists of length $n-2$ from S . We must show for each $a = (a_1, \dots, a_{n-2}) \in S^{n-2}$ that exactly one tree T with vertex set S satisfies $f(T) = a$. We can prove this by induction on n .

Advanced Graph Theory
Spanning Trees and Enumeration

So, Cayley's theorem says that for a given set S , which is basically of containing the elements from that label n . So, there are n raise to power n minus 2 different trees let us see the proof of this particular theorem. Now, this will hold for the number of vertices is 1. So, let us assume that n is greater than 2. So now, we will prove that this particular algorithm defines the bijection the algorithm which, we have seen earlier that will define a bijection function f from this set of trees with the vertex set, which is given as S to that particular set S raise to power n minus 2 of the list of the length n minus 2 from S , that we have already seen that, if let us say, tree is given how we can generate a Prufer sequence of length and minus 1? So, we must show that for each Prufer sequence a 1 to a

$n - 2$, which is a member of this S raise to power $n - 2$, that exactly only one tree with that vertex set satisfies this particular mapping function f and we can prove this by the induction on n .

(Refer Slide Time: 25:24)

Proof by Induction

Basis step: $n = 2$: ✓

- There is one tree with two vertices. The Prüfer code is a list of length 0, and it is the only such list. ✓

Induction step: $n > 2$: ✓

- Computing $f(T)$ reduces each vertex to degree 1 and then possibly deletes it. Thus every nonleaf vertex in T appears in $f(T)$. ✓
- No leaf appears, because recording a leaf as a neighbor of a leaf would require reducing the tree to one vertex. Hence the leaves of T are the elements of S not in $f(T)$. If $f(T) = a$, then the first leaf deleted is the least element of S not in a (call it x), and the neighbor of x is a_1 . ✓

Advanced Graph Theory
Spanning Trees and Enumeration

Let us take the induction on n , when n is equal to 2 there is only one tree with the vertices the Prufer code is a length $n - 2$, that is length 0 and it is the only such list. When induction hypothesis, let us assume n is greater than 2 now, computing $f(T)$ will reduce each vertex to a degree one; that means, degree of T minus 1 and then possibly it will get deleted, thus every non-leaf vertex in T appears in this particular f of T , that is in the Prufer sequence.

(Refer Slide Time: 26:37)

Contd...

- We are given $a \in S^{n-2}$ and seek all solutions to $f(T)=a$. We have shown that every such tree has x as its least leaf and has the edge xa_1 . Deleting x leaves a tree with vertex set $S' = S - \{x\}$. Its Prüfer code is $a' = (a_2, \dots, a_{n-2})$, an $n-3$ -tuple formed from S' .
- By the induction hypothesis, there exists exactly one tree T' having vertex set S' and Prüfer code a' . Since every tree with Prüfer code a is formed by adding the edge xa_1 to such a tree, there is at most one solution to $f(T) = a$. Furthermore, adding xa_1 to T' does create a tree with vertex set S and Prüfer code a , so there is at least one solution.

Now, here we also see that there is no leaf appears in S because, recording of the leaf as a neighbor of a leaf would require reducing the tree to one vertex. Hence, the leafs of tree are the elements of S not in $f T$. So, if $f T$ is equal to a then the first leaf gets deleted is the least element of S , which is not in a call it as x and the neighbor of x is a_1 . Now, we are given a which is an element in S raise to power n minus 2 and we see all the solutions to this particular mapping of T to a and we have shown that, every such tree has x as it is least leaf and has the edge $x a_1$ deleting x leafs a tree with the vertex set as prime, that we have already seen and it is Prufer code is a prime a_2 to a n minus 3 and n minus 3 tuple formed from S prime and so on. So, by the induction hypothesis there exist exactly 1 T tree prime having the vertex set S prime, that we have constructed and having the Prufer code a prime. Since, every tree with the Prufer code a is formed by adding an extra edge $x a_1$ to such a tree there is at most one solution to $f T$ is equal to a furthermore adding $x a_1$ to T prime does not create a tree with the vertex set x and the Prufer code a . So, there is at least one solution. So, that gets proved.

(Refer Slide Time: 27:50)

Corollary 2.2.4 count the trees by their vertex degrees.

- Given positive integers d_1, \dots, d_n summing to $2n-2$, there are exactly $\frac{(n-2)!}{\prod (d_i - 1)!}$ trees with vertex set $[n]$ such that vertex i has degree d_i , for each i .
- **Proof:** While constructing the Prüfer code of a tree T , we record x each time we delete a neighbor of x , until we delete x itself or leave x among the last two vertices. Thus each vertex x appears $d_T(x) - 1$ times in the Prüfer 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 subscripts to the copies of each i to distinguish them, then we are permuting $n-2$ distinct objects and there are $(n-2)!$ lists. Since the copies of i are not distinguishable, we have counted each desired arrangement $\prod (d_i - 1)!$ times, once for each way to order the subscripts on each type of label.

Advanced Graph Theory

Spanning Trees and Enumeration

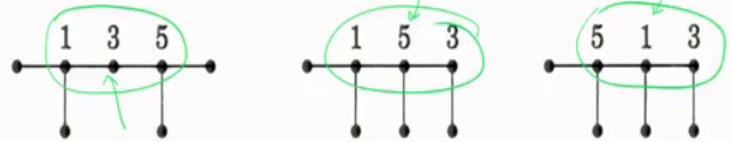
Now, corollary this particular corollary will count the trees by their vertex degrees. So, this is an application of a Cayley's or a Prufer code or Cayley's formula. So, given a positive integers d_1, d_2 and so on up to d_n they are nothing but the degrees of all n vertices and if we sum up these degrees to n minus 2 n minus 2 , there is exactly there is there are exactly n minus 2 factorial divided by d_i minus 1 factorials the multiplication of all for all i s. Let us see the proof. So, while constructing the Prufer code of a tree T we record x each time we delete the neighbor of x until, we delete x itself or leave x among the last 2 vertices thus each vertex appears how many times? This degree of x minus 1 times in the Prufer code therefore, we count the trees with these vertex degrees by counting the list of length n minus 2 , that is for each i have d_i minus 1 different copies of i so; that means, a particular label is appearing d_i minus 1 times. So, if we assign subscripts to this copies of i to distinguish them then we are permuting n minus 2 different distinct objects and there are n minus 2 factorial such list since, the copies of i are distinguishable we have counted each desired arrangement how many times d_i minus 1 factorial and multiply for all different i s once each way to order the subscript on each type of label.

(Refer Slide Time: 29:53)

Example

- **Trees with fixed degrees.** Consider trees with vertices $\{1,2,3,4,5,6,7\}$ that have degrees $(3,1,2,1,3,1,1)$, respectively. We compute $\frac{(n-2)!}{\prod (d_i-1)!} = 30$; The trees are suggested below

- Only the vertices $\{1,3,5\}$ are non-leaves. Deleting the leaves yields a subtree on $\{1,3,5\}$. There are three such subtrees, determined by which of the three is in the middle.



Advanced Graph Theory

Spanning Trees and Enumeration

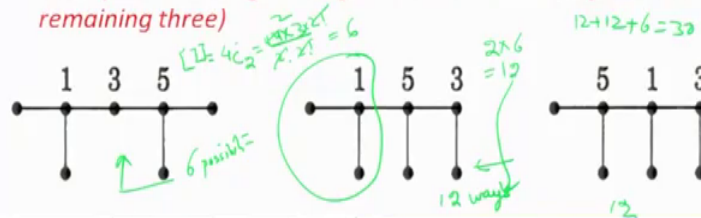
Let us see through an example. So, let us see the we are given the trees with the fixed degrees consider tree with the vertices 1, 2, 3 and so on up to 7 different vertices and the degrees of these vertices are given as 3 1 2 1 3 1 1 respectively. So, let us use this particular formula to find out how many different trees are there? So, that will be n minus 2 factorial divided by all the degrees. So, we can use in this particular equation if we apply. So, n means 7, 7 minus 2 is 5 factorial divided by all these degrees minus 1. So, first one is 3 minus 1 2 factorial then, is 0 factorial, 1 factorial, 0 one and again 2 factorial. So, 5 factorial divided by that becomes 30.

So, 30 different trees are possible. Now, out of these 30 trees we can see here that, only the vertices 1 3 and 5 1 3 and 5 they are non leaves they will be the internal notes. So, the deleting leaves will yield the tree on 1 3 5 and there are 3 such subtrees possible where, in the first tree we see that, in the center the 3 is there in the other one the center is 5 and the third one the center is 1, there are 3 different possibilities where, in these 5 1 3 and becomes the internal notes.

(Refer Slide Time: 31:47)

Contd...

- To complete each tree, we add the appropriate number of leaf neighbors for each non-leaf to give it the desired degree, There are six ways to complete the first tree (*pick from the remaining four vertices the two adjacent to vertex 1*) and twelve ways to complete each of the others (*pick the neighbor of vertex 3 from the remaining four, and then pick the neighbor of the central vertex from the remaining three*)



Advanced Graph Theory

Spanning Trees and Enumeration

So, let us count them there are 3 such subtrees by which, 3 is in middle and so on to complete each tree we add the appropriate number of leaf notes leaf neighbors to each of these non-leaf to give it the desired degree. So, there are 6 different ways to complete the first tree here. So, so 3 different labels are already used how many more are there? 4 different labels are remaining out of this particular 7 labels because, 3 are already used. So, 4 are remaining so; that means, if we choose out of 4 these 2 labels, which are going to be assigned to 1.

So, in turn it will and then 2 will be attached to the 5. So, $4C_2$ that becomes, so that becomes 6 different possibilities of this particular tree. Similarly, as per as this is concerned there are 12 possible ways here in this particular case, why? Because, 6 is choosing this possibilities now, both 2 the remaining 2 notes the remaining 2 different labels are not going to be attached to 5, but they are being distributed to 5 and 3. So, they are how many possibilities 2 possibilities are there 2 into 6 there are 12 different possibilities. Similarly, here, it is also having same thing that is 12 possibilities. So, total possibilities 12 plus 12 plus 6, that becomes 30. So, 30 different trees are being generated by this way.

(Refer Slide Time: 33:45)

Counting number of Spanning Trees in Graph

- We can interpret *Cayley's Formula* in another way.
- Since the **complete graph** with vertex set $[n]$ has all edges that can be used in forming trees with vertex set $[n]$, the number of trees with a specified vertex set of size n equals the number of spanning trees in a complete graph on n vertices.
- We now consider the more general problem of computing the number of spanning trees in any graph G . In general, G will not have as much symmetry as a complete graph, so it is unreasonable to expect as simple a formula as for K_n , but we can hope for an algorithm that provides a simple way to compute the answer when given a graph G .

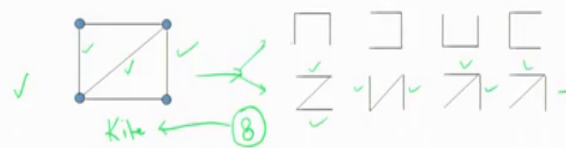
Now, counting the number of spanning trees in a graph now, we can interpret the Cayley's formula in another way since, the complete graph with the vertex set n has all the edges, that can be used in forming the trees with the vertex set n the number of trees with the specified vertex set of size n equal the number of spanning trees in the complete graph on n vertices. Now, consider more general problem of computing the number wise spanning trees in a graph in general G will not have much symmetry as a complete graph. So, it is unreasonable to expect a simple formula as K_n , but we hope for an algorithm that provide the simple way to compute the answer when a given graph is G .

(Refer Slide Time: 34:31)

Spanning Trees in Graphs 2.2.6

Example. A kite.

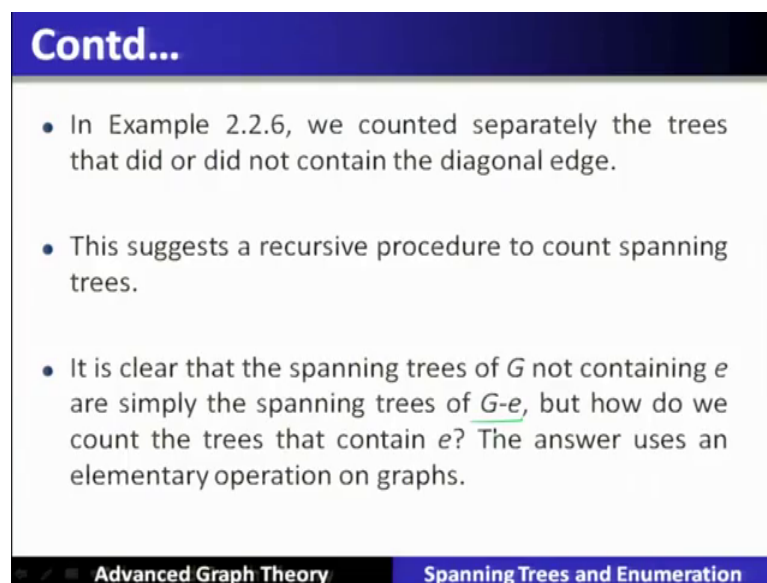
- To count the spanning trees, observe that four are paths around the outside cycle in the drawing
- The remaining spanning trees use the diagonal edge
- Since we must include an edge to each vertex of degree 2, we obtain four more spanning trees.
- The total is eight.



Let us, see in this particular example the same thing. So, if the graph is not complete graph let us, say a kite is given. So, this graph is called a kite the name of this graph is kite, kite is given and now we have to count how many spanning trees are there? Now, we if we observe that how many spanning tree? That means; we can form this spanning tree without this diagonal edge and we can form the spanning tree with diagonal edge there are only 2 possibilities.

So, if we form the spanning trees without this diagonal edge how many different possibilities are there? So; that means, these particular vertices are going to be attached with the edge an one set of vertices will not have an edge. So, there are only 4 different possibilities. Now, when the spanning trees when where the diagonal is used, that also is having 4 possibilities, why because, there are 2 ends who are not going to get the edges. So, you see that who are going to get the edges. So, this is one possibility, this is another possibility, this is another and so on. So, there are 4 different possibilities total 8 different spanning trees is possible from this particular graph, which is called a kite.

(Refer Slide Time: 35:57)



Contd...

- In Example 2.2.6, we counted separately the trees that did or did not contain the diagonal edge.
- This suggests a recursive procedure to count spanning trees.
- It is clear that the spanning trees of G not containing e are simply the spanning trees of $G-e$, but how do we count the trees that contain e ? The answer uses an elementary operation on graphs.

Advanced Graph Theory Spanning Trees and Enumeration

So, let us see in this example how using a systematic formula we are we can basically do this. So, it is clear that the spanning trees of graph G not containing e are simply the spanning trees of G minus e ; that means, if you remove that particular edge that, is nothing but a diagonal edge and when we basically include, that diagonal edge, so there are only 2 possibilities and we can count these 2 possibilities 4 and 4 that becomes 8.

(Refer Slide Time: 36:31)

Contraction 2.2.7

- In a graph G , **contraction** of edge e with endpoints u, v is the replacement of u and v with a single vertex whose incident edges are the edges other than e that were incident to u or v . The resulting graph $G \cdot e$ has one less edge than G .

Advanced Graph Theory Spanning Trees and Enumeration

Now, we for that we are going to define an operation, which is called a contraction of an edge e having the end point u, v . So, we can contract means, we can join on a same what on a same end and basically this edge is will be removed. So, for example, if we contract u and v as the same note all other edges will be preserved you can see all the other edges are preserved and this particular edge will be removed. So, this becomes $G \cdot e$, why? Because, it is a contraction we have not removed, but it is a contraction; that means, that edge is basically contracted this is called contraction operation.

(Refer Slide Time: 37:17)

Contd...

- In a drawing of G , contraction of e shrinks the edge to a single point, Contracting an edge can produce multiple edges or loops.
- To count spanning trees correctly, we must keep multiple edges.
- In other applications of contraction, the multiple edges may be irrelevant. The recurrence applies for all graphs.

Advanced Graph Theory Spanning Trees and Enumeration

So, to count the spanning tree correctly we must keep the multiple edges like here, this multiple keep the multiple edges resulting out of that contraction.

(Refer Slide Time: 37:35)

Recurrence

Proposition. Let $\tau(G)$ denote the number of spanning trees of a graph G . If $e \in E(G)$ is not a loop, then $\tau(G) = \tau(G - e) + \tau(G \cdot e)$ 2.2.8

Proof:

- The spanning trees of G that omit e are precisely the spanning trees of $G - e$.
- To show that G has $\tau(G \cdot e)$ spanning trees containing e we show that contraction of e defines a bijection from the set of spanning trees of G containing e to the set of spanning trees of $G \cdot e$.

Advanced Graph Theory Spanning Trees and Enumeration

So, with this we have basically come up with a recurrence equation and this recurrence equation will help us in finding out how many different spanning trees will be there for a given general graph G

(Refer Slide Time: 38:09)

Contd...

Proof:

- When we contract e in a spanning tree that contains e , we obtain a spanning tree of $G \cdot e$, because the resulting subgraph of $G \cdot e$ is spanning and connected and has the right number of edges.
- The other edges maintain their identity under contraction, so no two trees are mapped to the same spanning tree of $G \cdot e$ by this operation. Also, each spanning tree of $G \cdot e$ arises in this way, since expanding the new vertex back into e yields a spanning tree of G . Since each spanning tree of $G \cdot e$ arises exactly once, the function is a bijection.

Advanced Graph Theory Spanning Trees and Enumeration

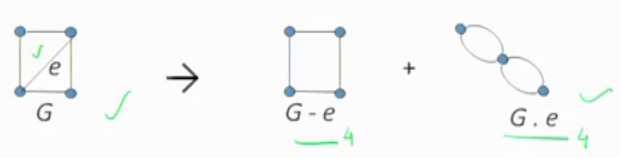
So, let us see the recurrence equation; that means, if we remove that particular middle edge e that, we have seen the diagonal edge when it is removed we have to count how

many are there? And then when we contract it then how many are there? And some of them will become that, total number of different spanning trees.

(Refer Slide Time: 38:12)

Example

- A *step in the recurrence*.
- The graphs on the right each have four spanning trees, so **Proposition 2.2.8** implies that the kite has eight spanning trees, Without the multiple edges, the computation would fail.



- We can save some computation time by recognizing special graphs G where we know $\tau(G)$, such as the graph on the right above.

Advanced Graph Theory Spanning Trees and Enumeration

So, let us take this particular example. So, if this is the graph which is a kite graph. So, when we remove this particular edge. So, this will be the graph without e and then this will be when we contract then, this will be graph when we have contracted the edge and now, if we count how many are the possibilities in this particular graph G minus e ? We have seen there are 4 possibilities and how many possibilities are there here? Again 4 4 plus 4 will become 8.

(Refer Slide Time: 38:45)

Remark 2.2.10

- If G is a **connected loopless graph** with no cycle of length at least 3, then $\tau(G)$ is the product of the edge multiplicities. A disconnected graph has no spanning trees.
- We cannot apply the recurrence of Proposition 2.2.8 when e is a loop. For example, a graph consisting of one vertex and one loop has one spanning tree, but deleting and contracting the loop would count it twice.
- Since loops do not affect the number of spanning trees, we can delete loops as they arise.

Advanced Graph Theory

Spanning Trees and Enumeration

So, basically here, the importance of the loop is not going to effect the number of spanning trees. So, if we can delete the loop, which is there in before we start then, also the number of spanning trees count will be correct. So, that is why? We consider always the connected loop less graph for computing the recurrence equation.

(Refer Slide Time: 39:09)

Contd...

- Counting trees recursively requires initial conditions for graphs in which all edges are loops.
- Such a graph has one spanning tree if it has only one vertex, and it has no spanning trees if it has more than one vertex.
- If a computer completes the computation by deleting or contracting every edge in a loopless graph G , then it may compute as many as $2^{e(G)}$ terms.
- Even with savings from **Remark 2.2.10**, the amount of computation grows exponentially with the size of the graph; this is impractical.

Advanced Graph Theory

Spanning Trees and Enumeration

(Refer Slide Time: 39:11)

Matrix Tree Theorem (Krichhoff)

- Another technique leads to a **much faster computation**.
- The **Matrix Tree Theorem**, implicit in the work of **Krichhoff [1847]**, computes $\tau(G)$ using a determinant.
- This is much faster, because determinants of n -by- n matrices can be computed using fewer than n^3 operations. Also, Cayley's Formula follows from the Matrix Tree Theorem with **$G = K_n$** , but it does not follow easily from Proposition 2.2.8.

Advanced Graph Theory

Spanning Trees and Enumeration

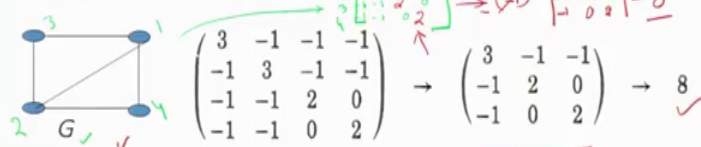
Now, we are going to see another very generic technique, and which is much faster in compare to the previous one and this method is called a matrix tree theorem or it is given by the Kirchhoff. So, basically for a given graph instead of computing the recurrence equation, we will apply this matrix tree theorem and this particular algorithm, which basically uses to compute the determinant and this computation of determinant of the adjacency matrix is basically quite efficient hence, this particular algorithm will compute the number of spanning trees for a very large values of n a much faster method. So, let us see how this all can be done.

So, this matrix tree theorem is much faster because, the determinants of n by n matrix can be computed using fewer operations than n^3 operations compare to the previous methods also the Cayley's formula follows from the matrix tree theorem, when G is equal to K_n , but it does not follow easily from this proposition that we have seen earlier.

(Refer Slide Time: 40:22)

Example: A Matrix Tree computation. 2.2.11

- **Theorem 2.2.12** instructs us to form a matrix by putting the vertex degrees on the diagonal and subtracting the adjacency matrix. We then delete a row and a column and take the determinant. When G is the kite of Example 2.2.9, the vertex degrees are 3,3,2,2. We form the matrix and take the determinant of the matrix in the middle. The result is the number of spanning trees!



- Loops don't affect spanning trees, so we delete them before the computation, The proof of the theorem uses properties of determinants.

Advanced Graph Theory

Spanning Trees and Enumeration

Let us take the example. So, in this example how this particular method will quickly compute that, we are going to see. Now, let us see that this particular graph is given and we want to find out how many different spanning trees are there? We have already seen using the recurrence now, we have to see this particular matrix tree computation how it does it? So, what we have to do? We have to find out first we have to draw a graph some matrix, that is called adjacency matrix of this particular graph and let us take this particular vertex as 1.

So, here this particular vertex 1 2, then 3 and 4 these are number 1 2 3 4 1 2 3 4. Now, we will fill it up with the adjacency equation; that means, whenever there is an edge we will place 1 if it is not then 0. So, between 1 and 2 we are going to place 1 1 and 4 we are going to place 1 1 and 3 we are going to place 1 and this becomes 0. Similarly, as per as 2 is concerned 2 and 3, 2 and 3 there is an edge 2 and 4 there is an edge 2 and 1 there is an edge 2 and 2 there is no edge. Now, as far as 3 is concerned 3 and 1 is an edge 3 and 2 there is an edge and all other is not having an edge.

Now, note number 4 2 is having an edge, 1 is having an edge and all others are not having an edge. So, this is the adjacency matrix of this particular graph. Now, what we are going to do is? We have to multiply these values by minus 1. So, they becomes minus wherever this symbols are there they becomes, minus next thing is we are going to instead of 0s in the diagonal we have to place the degrees of that particular note. So, for example, the degree of 1 is 3. So, in the diagonal instead of 0 we will write down 3 over here, similarly as per as the degree of note 2 is 3. So, instead of 0, we are going to

basically put 3 similarly, the degree of 3 is 2 instead of 0 we are going to put 2 and similarly, the degree of 4 is 2.

So, instead of 0 we are going to place 2 out of it. Now, then we are going to compute the determinant or a cofactor of this particular matrix. So, this particular way for example, minus 1 1 plus 1 and then, basically we have to find out this value, that is 3 minus minus 1 2 0 minus 1 0 2 we have to compute this value now, this becomes plus 1. So, that will go out now as per as this is concerned this you can compute easily, and that will be basically the value becomes 8. So, 8 we have seen this particular graph as given 8 using recurrence.

Now, this is very quiet efficient why because every big graph also we can convert in a form of the adjacency matrix and then, we will multiply by minus 1 and then, all the different degrees of the vertices are being placed here on the diagonal and then, we will compute the determinant or of this particular matrix, and that determinant will give the number of trees possible here, in that particular graph and that is called matrix tree computation.

(Refer Slide Time: 44:31)

Matrix Tree Theorem 2.2.12

Theorem: Given a loopless graph G with vertex set v_1, \dots, v_n , let a_{ij} be the number of edges with endpoints v_i and v_j . Let Q be the matrix in which entry (i, j) is $-a_{ij}$ when $i \neq j$ and is $d(v_i)$ when $i = j$. If Q^* is a matrix obtained by deleting row s and column t of Q , then $\tau(G) = (-1)^{s+t} \det Q^*$

Advanced Graph Theory Spanning Trees and Enumeration

So, let us see the theorem statement of a matrix tree theorem says that, given a loopless graph G with a vertex set v_1 to v_n , let a_{ij} be the number of edges with the end point v_i and v_j let Q be the matrix in, which the entry of ij is minus of a_{ij} when i is not equal to j ; that means, there is an edge. So, we have to put minus of a_{ij} and whenever, i is equal

to j ; that means, the diagonal element that we have talking about there, we have to place the degree of that particular vertex. Now, if Q star is a matrix which is obtained by deleting the row and the column of T of Q that we have already seen; so, we are going to get this cofactor that is minus 1 times S plus T that, we have already done types the determinant of Q .

So, this is how we are going to establish the matrix tree formula, matrix tree theorem? And we can solve this particular problem of finding the spanning tree on a given graph very efficiently using this particular matrix tree theorem.

(Refer Slide Time: 45:57)

Conclusion

- In this lecture, we have discussed the Enumeration of trees, Cayley's formula, Prüfer code, Algorithm for generation of Prüfer Sequence from Labelled Tree and Generation of Labelled Tree from Prüfer Sequence, Spanning trees in graphs, Matrix tree computation and Matrix tree theorem.
- In upcoming lecture, we will discuss the minimum spanning tree and shortest paths.

Advanced Graph Theory Spanning Trees and Enumeration

Conclusion, in this lecture we have discussed the enumeration of trees Cayleys formula, Prufer code algorithm for generating the Prufer sequence from Labeled tree, generation of Labeled tree from the Prufer sequence spanning trees in the graph matrix tree, computation matrix tree theorem. In upcoming lectures, we will discuss minimum spanning tree and shortest path.

Thank you.