Tree Counting Notes

18.310, Fall 2007, Prof. Peter Shor

(adapted from Jacob Green's OCW notes)

Preliminary version

1 Introduction

We will now consider two kinds of counting problems. The general form of the first kind of question we shall examine is: *If we define some kind of structure, which has size N, how many other structures of size N are there?*

Here are some questions of this form:

1. How many subsets of an M-element set are there of size N?
2. How many graphs are there on V vertices with N edges?
3. How many trees are there with N vertices?
4. how many trees on N vertices have exactly k leaves?

Another kind of question arises when there is some sort of symmetry among the structures we want to consider. We say that two structures have the same *pattern* when one can be gotten from the other by a symmetry operation (more on this in the next set of notes). We can then ask how many different patterns of structures we can have with certain given parameters.

Thus, for example, we can ask: *how many different patterns of graphs with N edges on V vertices are there?* In this case, since the N edges are in fact just pairs of vertices, we can permute the vertices in these pairs as our symmetry operation. This changes the labels on the vertices of our graph, but does not change the structure of the graph.

There is a fundamental difference between these two kinds of questions. One is counting instances of a structure; the other is counting patterns of structures. We shall look at some simple cases involving graphs to illustrate these differences more clearly.

We will look at some ordinary counting problems, then consider how we represent symmetry operations, and then some pattern counting problems.

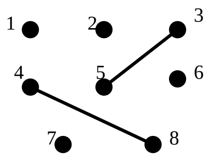
Of the counting questions listed at the beginning of the section, the first two are straightforward. The answer to question 1 is a basic result from combinatorics. The number of subsets of an M element set of size N is the binomial coefficient $\binom{M}{N}$, which is $\frac{M!}{N!(M-N)!}$.

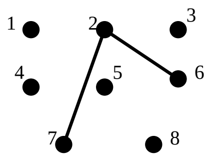
To answer question 2, we begin by considering the cases of $N = 1$ and $N = 2$, and comparing the number of graphs we get to the number of patterns we get. We first look at how many graphs on V vertices there are with one edge. Our edge will be between some two vertices in V, so this is a question of picking an unordered set of size 2 from a set of size V. The answer is the binomial coefficient $\binom{V}{2}$. Some simple algebra will show you that this is equal to $V(V-1)/2$. As far as patterns are concerned, all of these graphs have the same pattern: that of a single edge.

Similarly, there are

$$\frac{\frac{V(V-1)}{2}(\frac{V(V-1)}{2}-1)}{2}$$

different graphs on V vertices with two edges. On the other hand, there are only two patterns: either the two edges can be disjoint, or they can form a path of length two. So the answer to the pattern question for $N=2$ is 2. Here is an example of the two patterns for $V=8$.



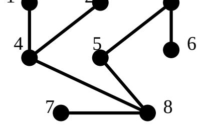


The number of graphs on V vertices and N edges, then is the number of ways of picking N edges out of the set of all possible edges (which has size $V(V-1)/2$). Thus, it is the binomial coefficient $\binom{V(V-1)/2}{N}$.

We now apply some of these ideas to trees in order to answer questions 3 and 4.

2 Counting Trees

Before looking at trees, let us recall what they are. A tree is a connected graph without any cycles. That is, there is a path from any vertex to any other, but no paths from a vertex to itself that do not use any edge more than once. Here is an example of a tree.



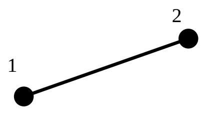
We will say that an "empty graph" is a graph on V vertices with no edges. The empty graph has $|V|$ (the number of vertices) connected components. Each edge that can be added to a graph G provides a path from one of its endpoints to the other. If there was already a path between these vertices (so that they were in the same connected component), then the new edge completes a cycle, and we will not have a tree.

Otherwise, each new edge joins two previously unconnected components of G into one, so that after $|V| - 1$ edges are added to the empty graph, we will have a tree.

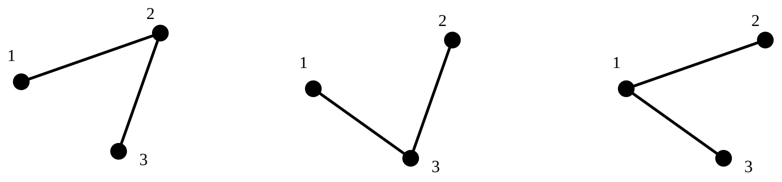
Thus, every tree on n vertices has $n - 1$ edges. In fact, we could have defined trees as connected graphs with $n - 1$ edges, or as graphs with $n - 1$ edges without cycles. In other words, any two of the three properties: $n-1$ edges, connected, and no cycles, implies the third.

We now ask: *how many trees are there on n vertices?* We can guess a formula by looking at the answer for small values of n.

It is clear that there is only one tree with two vertices: $\{\,(1,2)\,\}$:

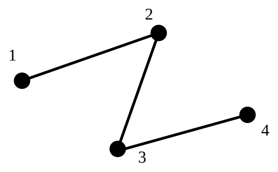


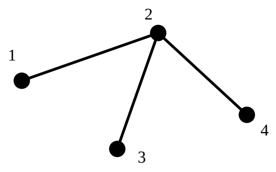
With three vertices, all trees are paths of length two. There are three of them: namely $\{\,(1,2),\ (2,3)\,\}$, $\{\,(1,3),\ (2,3)\,\}$, and $\{\,(1,2),\ (1,3)\,\}$:



With four vertices: there are two patterns of trees: a path of length three and a "claw" consisting of one vertex linked to all three of the others, as in $\{\,(1,2),\ (1,3),\ (1,4)\,\}$. Here is an example of one of each pattern:

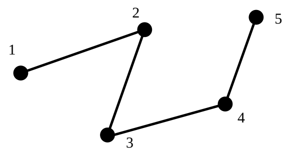
There are four possible clasws, one with each vertex as center. For the paths, there are $\binom{4}{2}$, or six, pairs of endpoints for the paths, and 2!, or 2 ways to arrange the

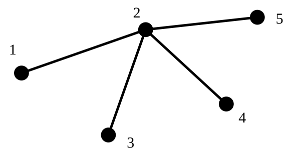


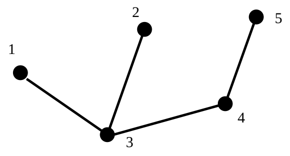


middle vertices of each of these, giving us twelve possible paths altogether. Adding together the claws and the paths we get a total of 16 possible trees on four vertices.

With five vertices, there are three paterns: a path of length 4, a claw, and a Y (whose lower part is a path of length two):







There are 5 possible claws, one for each vertex. There are $\binom{5}{2}\cdot 3!$, or 60, paths (since there are $\binom{5}{2}$) possible pairs of endpoints and 3! ways to arrange the intermediate vertices for each of these pairs of endpoints). There are also $\binom{5}{2}\cdot 3!$, or 60, Y trees, since there are $\binom{5}{2}$ ways to choose the top vertices of the Y, and 3! ways to arrange the rest. This gives us $5 + 60 + 60 = 125$ total trees on 5 vertices.

We therefore have the first terms of the sequence. If we define the number of trees on n vertices to be $F(n)$, we have

$$F(2) = 1, \quad F(3) = 3, \quad F(4) = 16, \quad F(5) = 125.$$

Do you see a pattern? $125 = 5^3$, and taking our cue from this, we see that so far $F(n) = n^{n-2}$. Can we prove this hypothesis?

3 Proving that the number of trees on n vertices is n^{n-2}

One way to count these trees is to define another structure whose size we know to be n^{n-2}, and then show that we can assign a unique tree to each of them, and vice versa. This is known as finding a *bijection*, and is a common technique in combinatorics.

What then does n^{n-2} count.

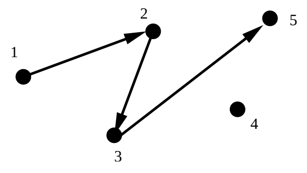
Suppose we have n object, O_1, O_2, $\ldots$, O_n, and we pick one. There are n ways to do this. If we throw the object we pick back and pick again, then there are n

possible outcomes as well. Thus, if we pick objects independently in this manner a total of $n-2$ times, there will be n^{n-2} different ways to do this.

Another way of looking at these n^{n-2} things is as *functions from a set of size $n-2$ to a set of size n*. For each of the $n-2$ possible inputs to the function f, we must choose one of n possible outputs. There are thus n^{n-2} possible functions altogether.

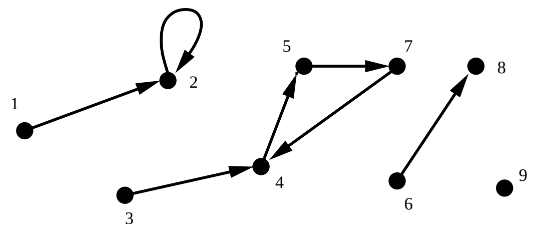
We will now describe a given sequence of choices (or function) graphically.

In order to do so, we need to make one new definition: that of a directed graph. A *directed graph* has the same elements as a regular graph; namely a set of vetices and edges. There is one important difference: in a directed graph, the edges have a direction, meaning that they point from one vertex to the other. Therefore, unlike in a normal graph, when we write the pairs of vertices that represent an edge, the order matters. So the edge (j,k) represents an edge pointing from vertex j to vertex k. Here is a drawing of the directed graph $\{\,(1,2),\ (2,3)\,(3,5)\,\}$ on the vertices $\{1,2,3,4,5\}$.



Let $f(j) = x_k$ indicate that we chose the kth object for our jth choice. Then we can draw a directed graph and put in a directed edge (j,k) from vertex j to vertex k, for each such choice. Remember that we make $n-2$ choices, and each choice can be an of the n objects.

In the example pictured below, we have $f(1) = 2$, $f(2) = 2$, $f(3) = 4$, $f(4) = 5$, $f(5) = 7$, $f(6) = 8$, $f(7) = 4$. This graph corresponds to choosing O_2 first, O_2 second, then O_4, O_5, O_7, O_8, O_4.



The directed graph that we form from our $n - 2$ choices in this way will have the following properties:

1. There will be exactly one edge directed *from* each vetex with index $\leq n - 2$, and none from the last two vertices. (This is because there are only $n - 2$ choices, and each edge is directed from the vertex of the choice number, and towards the vertex of the object selected.)

2. It can have directed cycles or even loops (since one could pick the jth object on the jth choice)

3. Each connected component contains at most one cycle. Each component will either contain vertex $n - 1$ or vertex n, or a cycle. (This is because if you follow the directed path from any vertex, you will either end up at vertex $n-1$ or at vertex n, which have no edges leading away from them, or you will end by going around some cycle.)

Our plan is to make each graph into a unique tree in a reversible way.

Now, a tree is different from one of our graphs in the following respects. First, a tree is an undirected graphs. We can change this by introducing a direction to each edge of the tree, namely, towards the last vertex in V, vertex n. If we do so, every vertex other than the last will have exactly one edge directed away from it.

The difference between our graphs and trees is then the following:

a. Our graphs have no edge directed from vertex $n-1$ while a directed tree does.

b. Our graphs can have loops and directed cycles while trees cannot.

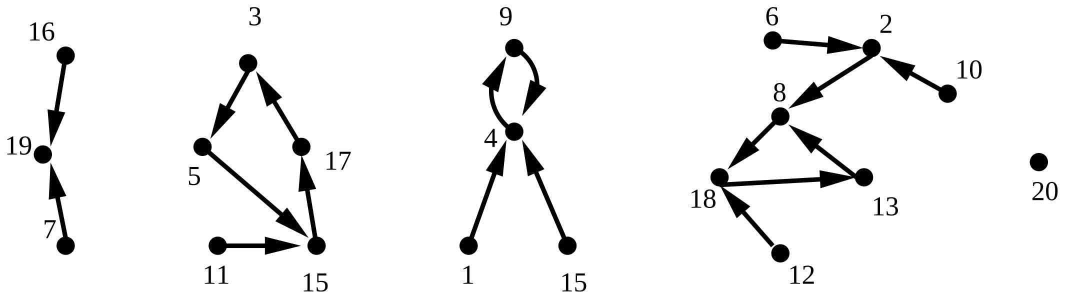
c. There may be no edge directed into vertex n in one of our graphs, but there must be at least one in every directed tree (since every vertex in a tree must have at least one edge, and there is no edge directed *from* vertex n).

d. Our graphs have $n - 2$ edges while trees have $n - 1$ of them.

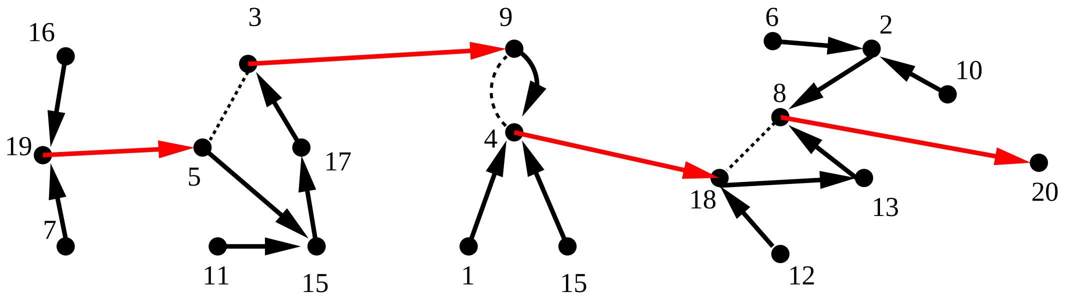
We will convert one of our graphs into a tree by adding to it a directed path from vertex $n-1$ to vertex n that passes through and neutralizes (eliminates) every cycle of our graph.

This leaves us with three questions: *How do we order the cycles on the path? How do we pass through a cycle to neutralize it? Moreover, how do we reverse this process to regain our graph uniquely from the tree it creates?*

We label each cyle in one of our graphs by the smallest index of the vertices in it. For example, the cycle $\{(2,3), (3,5), (5,2)\}$ gets the label 2. We then order the cycles by these labels and we will traverse them in ascending order. Here is a graph in which the cycles are labeled 3, 4, 8. We're ignoring vertices not in cycles in this labeling.



Here is how we neutralize a cycle. We have our path from $n-1$ to n enter the cycle at the vertex immediately after the label vertex of the cycle, and exit again at the label vetex. We then omit the edge that is directed from the label vertex to the vertex after it in the cycle. In the above example, we enter the first cycle at vertex 5 and leave the cycle at vertex 3, omitting the edge $(3, 5)$. (We show the new edges we have added in red).



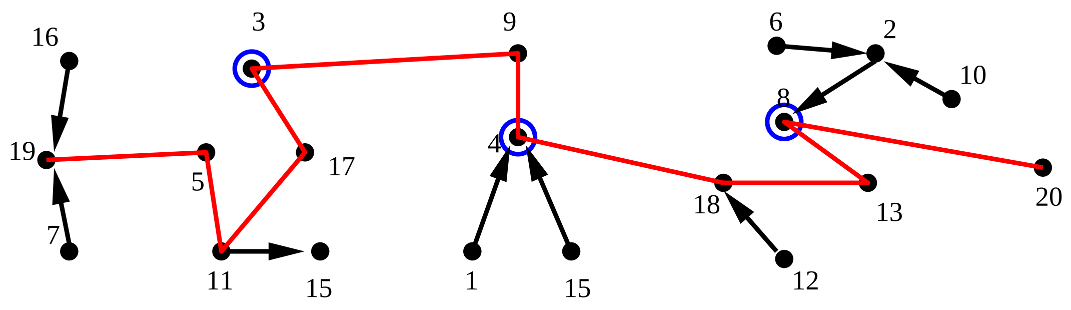
So, we will have a path that goes from vertex $n-1$, neutralizes all the cycles in order, and finally ends at vertex n.

Are we really guaranteed to get a tree after introducing the path from $n-1$ to n and neutralizing the cycles? Well, if we look at the procedure outlined above, we see that the graph that results has no cycles and $n-1$ edges, which as we said in the previous section, defines a tree.

Now, how do we get back from a tree to one of our function graphs in a well-determined fashion? Notice that the smallest vertex index on the path from $n-1$

to n in the resulting tree will mark the end of the first cycle we neutralized. The smallest index after that on the path marks the end of the second cycle, and so on. This means that, given a tree, we can examine the path in it from vertex $n-1$ to n, and find the smallest vertex on it. We know that the first edge of our path connects to the cycle with the smallest index, at the vertex that the label vertex is directed towards. This means that we can close our first cycle by simply drawing an edge directed from the smallest vertex in the path to the second vertex in the path. We know that the path leaves this cycle and goes to the second cycle, entering it at the vertex right after the label vetex. This tells us which edge to use to complete this cycle as well. We continue in this manner until we have reconstructed all of the cycles. We then omit every edge of the path that is not in one of our cycles. This gives us back our original graph.

If we look at the tree that we constructed in our last example, we can illustrate this process.



We look at the path from $n-1$ to n, and find the smallest vertex on it, 3 in this case. It's circled in the figure. We then find the smallest vertex on the path between this vertex and n, and so on. These circled vertices are the smallest labels in their cycles, and we break the path from $n-1$ to n after these vertices, and put an edge from these vertices to the vertex after the previous circled vertex (or $n-1$ for the case of the first cycle. You can check that this process will always give you back the directed graph you started with.

Thus, we have shown that every set of $n-2$ choices from n objects can be uniquely represented as a graph, that in turn can be uniquely represented as a tree, and vice versa. This gives us our bijection between the number of trees on n vertices and the number of ways to make $n-2$ choices of n objects, and thus concludes the proof.

4 Counting trees with given degrees at the vertices

There is an extension to the tree counting theorem which is also quite remarkable. Suppose we count trees, and take into account the degrees of the vertices. Let the degree of the vertex labeled i of a tree T be $d(i, T)$. Now, for a tree T, associate with it the monomial

$$x_1^{d(1,T)-1} x_2^{d(2,T)-1} \dots x_n^{d(n,T)-1}.$$

This monomial will have degree $n-2$, since for any graph, the sum of the degrees of the vertices is twice the number of edges, which is $2n-2$ for a tree. Now, suppose we add up these monomials for all the trees on n vertices. What do we get? It turns out that

$$(x_1 + x_2 + \dots + x_n)^{n-2} = \sum_T \prod_{i=1}^{n} x_i^{d(i,T)-1}.$$

where the sum is over all labeled trees T on n vertices.

This can be proved by the same bijection proof that we gave in the previous section. If we let the number of inputs mapped to i be $c(i)$, and associate with a function f the monomial $\Pi_i x_i^c(i, f)$, then a simple counting argument shows that for the functions discussed in the previous section, we have

$$(x_1 + x_2 + \dots + x_n)^{n-2} = \sum_f \Pi_i x_i^{c(i,f)}.$$

Then to prove the formula for trees, all we have to do is look at our bijection more carefully and show that it takes a function to a tree with $d(i) = c(i) + 1$. This isn't hard to do. The important point is that have only added more edges to nodes $n-1$ and n, which were the nodes that didn't have edges going out of them.

We will prove this formula in a different way just for fun (thus also giving another proof of the tree counting theorem), and also to illustrate what techniques can be used with generating functions.

Suppose we set one of the variables, say x_n to 0 in the formula. What happens? On the left hand side, we get

$$(x_1 + x_2 + \dots + x_{n-1})^{n-2}.$$

On the right hand side, we get our sum over all trees which do not contain the variable n in the product $\prod_{i=1}^{n} x_i^{d(i,T)-1}$. This happens only if $d(n, T) = 1$; that is, when vertex n is a leaf. How can we evaluate the sum over the trees in which vertex n is a leaf. Consider what happens when we remove vertex n: we get a tree T' on vertices 1, 2, …, $n-1$. This lets us use our induction hypothesis. Now, for every tree T' on vertices $1, 2, \dots, n-1$, we can add vertex n as a leaf hanging off any of these $n-1$ vertices, say vertex i. If we hang it off vertex i, we will increase the degree of vertex i by 1, multiplying our polynomial by x_i. So if we sum the effects of hanging it off all the other vertices, for each tree T' on vertices $1, 2, \ldots, n-1$, we multiply its monomial by $(x_1 + x_2 + \ldots + x_{n-1})$ to get its contribution in the sum on trees on n vertices. But this is exactly what we get when we set x_n to 0 in the formula we're trying to prove. Thus, if we know our formula is true for trees on $n-1$ nodes, we have proved that it is true for trees on n nodes, when we just restrict to looking at trees where vertex n is a leaf.

However, by symmetry, the same argument would work trees where any given vertex is a leaf. And every tree has some leaf, so we have proved our formula.

5 Counting subtrees of a graph

Suppose we want to count trees whose edges all belong to a given graph G. These are called *spanning trees* of G. Notice that for G to have any trees as subgraphs, it has to be connected. If G is the complete graph K_n, we already know the answer, it is n^{n-2}. Suppose G is the five-vertex graph $\{(1,2),(1,3),(1,4),(1,5),(2,3),(3,4),(4,5)\}$ shown below. How do we count these trees?

What we do is first make a matrix M with a -1 in position if (i,j) if there is an edge from vertex i to j in G. The matrix $-M$, with 1's instead of -1's, is generally called the *adjacency matrix* of G. Here is M for our above G.

$$\begin{pmatrix} 0 & -1 & -1 & -1 & -1 \\ -1 & 0 & -1 & 0 & 0 \\ -1 & -1 & 0 & -1 & 0 \\ -1 & 0 & -1 & 0 & -1 \\ -1 & 0 & 0 & -1 & 0 \end{pmatrix}$$

We next put the elements on the diagonal which will make all the rows and columns add up to 0. This means the the entry (j,j) is equal to the degree of vertex j.

$$\begin{pmatrix} 4 & -1 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 \\ -1 & 0 & -1 & 3 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{pmatrix}$$

This matrix is singular, so it has determinant 0. Next, we delete one row and column, and take the determinant of the resulting matrix. For this matrix, we get 21. And there are indeed 21 trees contained in the graph G (CHANGE TO SMALLER EXAMPLE?)

How do we prove this? We'll do it the same way as we did earlier: we'll count trees which have a given degree sequence on their vertices. We do this by replacing the -1's in the ith column by x_i's, and adding the appropriate sum of x_i's on the diagonal to make the row sums 0.

$$\begin{pmatrix} x_2+x_3+x_4+x_5 & -x_2 & -x_3 & -x_4 & -x_5 \\ -x_1 & x_1+x_3 & -x_3 & 0 & 0 \\ -x_1 & -x_2 & x_1+x_2+x_4 & -x_4 & 0 \\ -x_1 & 0 & -x_3 & x_1+x_3+x_5 & -x_5 \\ -x_1 & 0 & 0 & -x_4 & x_1+x_4 \end{pmatrix}$$

The determinant of this matrix is 0, as all the columns sum to 0. Suppose we delete the ith row and the ith column. The determinant of the matrix gives us a polynomial whose degree is $n-1$, so it can't be the sum of the monomials

$$x_1^{d(1,T)-1} x_2^{d(2,T)-1} \dots x_n^{d(n,T)-1},$$

as all these monomials have degree $n-2$. What is this determinant? It turns out to be x_i times the sum of monomials associated with all the subtrees of G. How can we prove this? Again, we can set some $x_j = 0$ and prove it by induction by just considering the trees where j is a leaf. I'll let you work out the details for yourself — it's not very hard. The key step is noticing that if you set some x_j to 0, the only non-zero entry in the jth column is the sum of x_k over the neighbors k of j. These are the only places you can attach k as a leaf of the tree.