Lecture 3: Common graph families, trees, and Cayley's theorem · 1MA170

Vilhelm Agdur[1]

[1] vilhelm.agdur@math.uu.se

2 November 2023

We start by introducing a few named families of graphs. Then we introduce the class of *trees*, and prove some results about them, such as that they have leaves.[2] The main result is Cayley's theorem, which counts the number of labelled trees on n vertices.

[2] Contradicting what our own eyes can see outside in the November weather.

Common graph families

We warm up today by giving names to some common families of simple graphs that we will see reappearing throughout the course. They are illustrated in Figure 1.
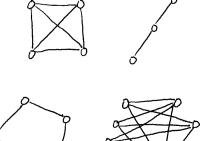

Figure 1: Four graphs: K_4, P_2, C_5, and $K_{3,2,1}$.

1. The complete graphs on n vertices, denoted K_n. These contain all the $\binom{n}{2}$ potential edges. These are also called *cliques* when we see them as subgraphs of a bigger graph.

2. The paths of length ℓ, denoted P_ℓ. If we take the set $\{0, 1, \ldots, \ell\}$ to be our vertex set, the edges are precisely of the form $\{i-1, i\}$ for $i \in [\ell]$.[3]

[3] This is another notation you might not have seen before: For an integer n, we write $[n]$ for the set $\{1, 2, \ldots, n\}$

3. The cycle graphs on n vertices, denoted C_n. We can think of these as a path of length $n-1$ with an extra edge joining the first and last vertex.

4. The complete bipartite graphs on $a + b$ vertices, denoted $K_{a,b}$. These have as vertex set the disjoint union of two sets L and R,[4] with $|L| = a$ and $|R| = b$, and there is an edge between v and w whenever $v \in L$ and $w \in R$. When we see these graphs as subgraphs of a bigger graph, we sometimes also call them *bicliques*.

[4] Think of these as the "left" and "right" vertices.

5. Generalizing the complete bipartite graphs, the complete multipartite graph on r parts with sizes $a_1, a_2, \ldots a_r$, denoted $K_{a_1, a_2, \ldots, a_r}$, has as its vertex set the disjoint union of r sets $V_1, V_2, \ldots, V_r$, where $|V_i| = a_i$, and there is an edge between two vertices whenever they are not in the same part. We can notice that when $r = 2$ this is a complete bipartite graph, and when all the parts are of size 1 this is a complete graph.

For most of these, it is obvious how many edges they will have. Let us state a lemma that shows how many the complete multipartite graphs have.

Lemma 1. *The complete multipartite graph* $K_{a_1, a_2, \ldots, a_r}$ *has* $\frac{1}{2} \left(n^2 - a_1^2 - \ldots - a_r^2 \right)$ *edges.*

Proof. We use the handshake lemma from the previous lecture. Since a vertex in V_i has one edge to every vertex not in V_i, it has degree $n - a_i$, and there are a_i such vertices. Thus we can compute that

$$2|E| = \sum_{v \in V} d_v = \sum_{i=1}^{r} \sum_{v \in V_i} d_v$$

$$= \sum_{i=1}^{r} \sum_{v \in V_i} (n - a_i) = \sum_{i=1}^{r} a_i (n - a_i)$$

$$= n \left(\sum_{i=1}^{r} a_i \right) - \sum_{i=1}^{r} a_i^2 = n^2 - \sum_{i=1}^{r} a_i^2$$

proving the claim. □

Corollary 2. *The complete bipartite graph* $K_{a,b}$ *has* $\frac{1}{2} \left(n^2 - a^2 - b^2 \right) = ab$ *edges.*

Trees

The main topic of this lecture is the so-called *trees*. Informally, they are just graphs that look like trees – though with this intuition we are drawing them upside-down.[5]

Definition 3. A *tree* is a graph $T = (V, E)$ that is both connected and contains no cycles.

Just like for graphs in general, there are countless variants of the notion of a tree. We can give the tree a root,[6] we can consider orderings of the vertices in various ways, and so on. None of this will actually be necessary in the course, however, so we skip defining these notions.

[5] Unless you are a computer scientist, in which case you draw them the right way up, I believe.

[6] Which is quite necessary for a biological tree.

Example 4. Trees appear in many places in various areas – one common place for them to appear is in the study of algorithms. Let's study the quicksort algorithm, and see how it can be represented as a tree. The algorithm sorts a list of numbers in ascending order, and it works as follows:
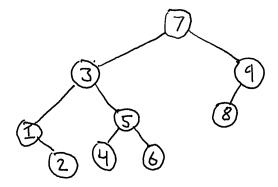
1. Fix an arbitrary pivot p element from the list.

2. Compare all non-pivot elements with the pivot, and put the ones that are smaller in a list we call L, and the ones that are larger in a list we call R.[7]

3. Apply the quicksort algorithm to L and R if they contain more than one element.[8]

4. Return the list LpR.

We can represent this algorithm with a rooted ordered binary[9] tree. This means the tree has one designated vertex we call the root, and every vertex has potentially a left child and a right child, where a "child" is a neighbour who is further from the root than themselves.

We create a tree from this algorithm by letting the pivot element be the root of the tree, and then recursively letting its left child be the root of of the tree quicksort gives us for the list L, and the right child be the root of the tree we get for the list R. This hopefully becomes clearer if we work through an example:

Consider the list $3, 7, 1, 4, 9, 8, 6, 2, 5$, and choose 7 as our pivot element. Then we get $L = 3, 1, 4, 6, 2, 5$ and $R = 9, 8$.

So, recursing, we now need to sort the list $3, 1, 4, 6, 2, 5$, and we pick 3 as our pivot element, getting a new $L = 1, 2$ and $R = 4, 6, 5$. If we pick 1 as our pivot in L, we get $L = \varnothing$ and $R = 2$. Picking 5 as out pivot in the previous R, we get $L = 4$ and $R = 6$.

Finally, we need to quicksort the list $9, 8$, where we can pick 9 as our pivot and get $L = 8$, $R = \varnothing$. The resulting tree of what we just did is drawn in Figure 2.
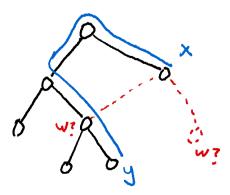
[7] This is of course not the most efficient way to *implement* this algorithm – the correct thing to do is to move elements around in a single list, since this can be done "in place" and so will be faster. But this is mathematically equivalent, and easier to phrase.

[8] Since we did not include the pivot element in either set, they are both strictly shorter lists than the one we started with, so this recursion will terminate.

[9] Binary in the sense of a computer scientist, since this is after all an algorithm. A mathematician would have a slightly different definition of binary tree.

Figure 2: A tree gotten from quicksorting a list of integers in our example.

Having seen this example of a place where trees appear, let us start proving some properties of trees.

Lemma 5. *Every finite[10] tree with at least 2 vertices contains at least two vertices of degree 1. Such vertices are called leaves.*

Proof. Let T be a finite tree on at least two vertices. Consider a path $P = xe_1x_1e_2\ldots y$ of maximum length in T. Assume for contradiction that one of x and y – w.l.o.g.,[11] let's say x – has degree at least 2.

Then x has a neighbour w different from x_1. There are two cases, which are also illustrated in Figure 3:

1. w is not already a vertex on the path P. Then we could just extend the path to include w, contradicting our assumption that P was maximal.

2. w is already a vertex on the path P. However, this must mean there is a cycle in the graph – we can start walking along T from x until we reach w, and then loop back along the edge from w to x – contradicting our assumption that the graph is a tree.

So neither case is possible, and so we conclude that $d_x = d_y = 1$, proving the lemma. $\qquad\square$

We also get a very simple formula for the number of edges of a tree:

Lemma 6. *Any tree on n vertices has $n - 1$ edges.*

Proof. We prove this by induction in the number of vertices. The base case is the simple graph on a single vertex, which clearly has one vertex and zero edges, satisfying the theorem.

Now, suppose T is a tree on $n > 1$ vertices. By Lemma 5, it has at least two vertices of degree 1. Pick one of them, say x, and remove it and the edge it is incident to from the graph.[12]

This leaves us with a tree on $n - 1$ vertices, and so by our induction hypothesis it has $n - 2$ edges. So by removing a vertex and an edge,

[10]

Exercise 1. Why do we need the finiteness assumption?

[11] "without loss of generality".

Figure 3: The two cases we consider in the proof of Lemma 5, illustrated in dashed red lines, with the maximal path drawn in blue.

[12] So, formally, we are looking at the induced subgraph $T[V \setminus \{x\}]$.

we were left with $n - 1$ vertices and $n - 2$ edges – so clearly we must have started with n vertices and $n - 1$ edges, as the lemma claimed. □

We can actually give an exact count of the number of labelled trees on n vertices as well. We will state the theorem here, but we postpone the proof of it until a later lecture, when we will have seen the Kirchhoff matrix-tree theorem, which is a nice way of proving it.[13]

Theorem 7 (Cayley's formula). *There are n^{n-2} labelled trees on n vertices.*

Having seen all these results about trees, let us give a result giving a few equivalent characterisations of trees:

Theorem 8. *The following statements are equivalent:*

1. *T is a tree.*

2. *For any two vertices x, y of T there exists a unique path connecting x and y.*

3. *T is edge-minimal among connected graphs, that is, removing any edge from it makes it disconnected.*

4. *T is edge-maximal among cycle-free graphs, that is, adding an edge between any two vertices introduces a cycle.*

Proof. We will show that (1) implies (2), that (2) implies (3) and (4), and that both (3) and (4) imply (1), showing the theorem.

(1) $\Rightarrow$ (2): Assume T is a tree and x and y two vertices. Since T is connected, there is at least one path from x to y. If there was another distinct path, the union of the two paths would contain a cycle, which is impossible in a tree. Thus the path is unique.

(2) $\Rightarrow$ (3): Consider any edge $\{x, y\} \in E$. By (2), this must in fact be the *unique* path between x and y, so removing this edge would disconnect these two vertices and thus the graph.

(2) $\Rightarrow$ (4): Consider any two vertices $x, y \in T$. By (2), there is a unique path connecting them. Adding an edge $\{x, y\}$ would thus create a cycle – walk along the path from x to y and then head back along the new edge.

(3) $\Rightarrow$ (1): If T is edge-minimal among connected graphs, it is of course connected. So what we need to show to show that it is a tree is that it contains no cycles. So, assume for contradiction that it has a cycle – then we could delete an edge of this cycle without disconnecting the graph,[14] contradicting our assumption of edge-minimality.

(4) $\Rightarrow$ (1) : If T is edge-maximal among cycle-free graphs, it is in particular cycle-free, and so what we need to show in order to show

[13] In previous years, this course included a proof of this theorem using Prüfer sequences. However, that proof – and a different one using double counting – is also included in the course on combinatorics in this department. So, in the interest of keeping the contents of the courses disjoint, we give the proof using the matrix-tree theorem instead.
  If you are interested, and can read Swedish, you can find the other two proofs of this theorem in these lecture notes: `https://vagdur.github.io/Kombinatorik-1MA020/lecture8.pdf`.

[14] Why?

that it is a tree is that it is connected. So assume for contradiction that it is disconnected – then we could add an edge between two connected components without introducing any cycles, contradicing our assumption of edge-maximality.   □

Spanning trees

Recall from last lecture that a subgraph is said to be spanning if it contains all vertices of *G*. This notion is particularly interesting if the subgraph is a tree.
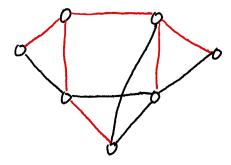

Figure 4: A graph *G*, with a spanning tree highlighted in red.

Definition 9. Let *G* be a graph, simple or multi. A *spanning tree* of *G* is a spanning subgraph that is a tree. One example is given in Figure 4.

Obviously, a graph that is disconnected cannot have any spanning trees – and for finite graphs, this condition is also sufficient.[15] For infinite graphs, the situation is somewhat hairier – the statement that *all* graphs have spanning trees is in fact equivalent to the axiom of choice!

Before proving this statement under the assumption of choice, let us state the form of the axiom of choice that we will be using.

[15]

Exercise 2. Give a proof of this that does not rely on the axiom of choice.

Lemma 10 (Zorn's lemma). *Let $(A, \leq)$ be a non-empty partially ordered set. A subset $C \subseteq A$ is a* chain *if for any two elements $c, c' \in C$, we either have $c \leq c'$ or $c' \leq c$. Assume that for every chain C in A there exists an upper bound $b \in A$, that is, an element such that $c \leq b$ for all $c \in C$. Then there exists a maximal element m of A, that is, an m such that $m \not\leq a$ for any $a \neq m$.*

Having stated this, we can now proceed to state and prove our theorem about spanning trees.

Theorem 11. *Assuming the axiom of choice, all (multi)graphs have a spanning tree.*

Proof. Let $G = (V, E)$ be some graph. We will apply Zorn's lemma with *A* as the set of all cycle-free spanning subgraphs of *G*, and $\leq$ as

the "is a subgraph of" relation.[16] This gives A the structure of a partially ordered set – and it is nonempty since the subgraph containing all the vertices and none of the edges is trivially spanning and cycle free.

So, let C be some chain in A, consisting of elements $H_i = (V, E_i)$ for $i \in I$,[17] and define

$$B = \left(V, \bigcup_{i \in I} E_i \right).$$

We want to show that B is an upper bound for the chain C – if we can do this, Zorn's lemma will apply and give us the theorem.

By construction, B is a spanning subgraph of G. We want to show that it is an element of A, so we need to show that it is also cycle-free. So, assume for contradiction that it contains some cycle consisting of edges $e_1, e_2, \ldots, e_r$. Each of these edges must then be contained in some H_i – let us say that e_j is contained in $H_{i(j)}$.

Since C is a chain, one of these graphs $H_{i(1)}, H_{i(2)}, \ldots, H_{i(r)}$ must contain all of the others – say it is H_j. Then this H_j must contain all of the edges $e_1, e_2, \ldots, e_r$, and so contain a cycle – but this is a contradiction, since H_j is an element of A and thus cycle-free.

Now, we need to show that B is indeed an upper bound of the chain C, in order for Zorn's lemma to apply. This, however, is easy – of course $E_i \subseteq \bigcup_{i \in I} E_i$, and so H_i is a subgraph of B for each i.

The conclusion of Zorn's lemma now gives us the existence of a cycle-free spanning subgraph that is maximal with respect to the subgraph ordering, which in this case means it is edge-maximal. So, by an analogous argument as in Theorem 8,[18] this maximal subgraph is in fact a tree, and so we have found our desired spanning tree. □

Let us close out this lecture by defining the notion we will be studying next lecture, namely the *complexity* of a graph.

Definition 12. Let G be a labelled graph. The *complexity* of G is the number of spanning trees of G, denote $t(G)$.

Example 13. Any tree of course has complexity 1, because its only spanning subgraph is the tree itself. A cycle graph on n vertices will have complexity n. Cayley's formula[19] tells us that the complexity of K_n is n^{n-2}. It turns out that the complexity of the complete bipartite graph $K_{a,b}$ is $a^{b-1}b^{a-1}$, but this is not an easy fact to prove.

Exercises

Exercise 3. We proved that all graphs, including infinite graphs, have a spanning tree using the axiom of choice in the lecture, and we stated that in fact the two are equivalent.

[16] Here we mean literal subgraph of, not just "isomorphic to a subgraph of" – we really need one to be a subset of the other.

[17] We really need the index *set I* here, not just integer indices – the cardinality of C could be enormous!

[18] We can't quite apply this theorem directly, because it assumes it is edge-maximal among *all* cycle-free graphs, not just among the cycle-free subgraphs of a fixed graph. In last year's lecture notes they just appeal to this theorem, but that seems to me to be subtly wrong.

[19] Which we admittedly haven't proven yet.

Prove the axiom of choice using the assumption that all graphs have spanning trees.