6.1200J/18.062J *Mathematics for Computer Science* Tuesday 2nd April, 2024
Massachusetts Institute of Technology, Spring 2024
Z. Abel, B. Chapman, E. Demaine revised Monday 1st April, 2024

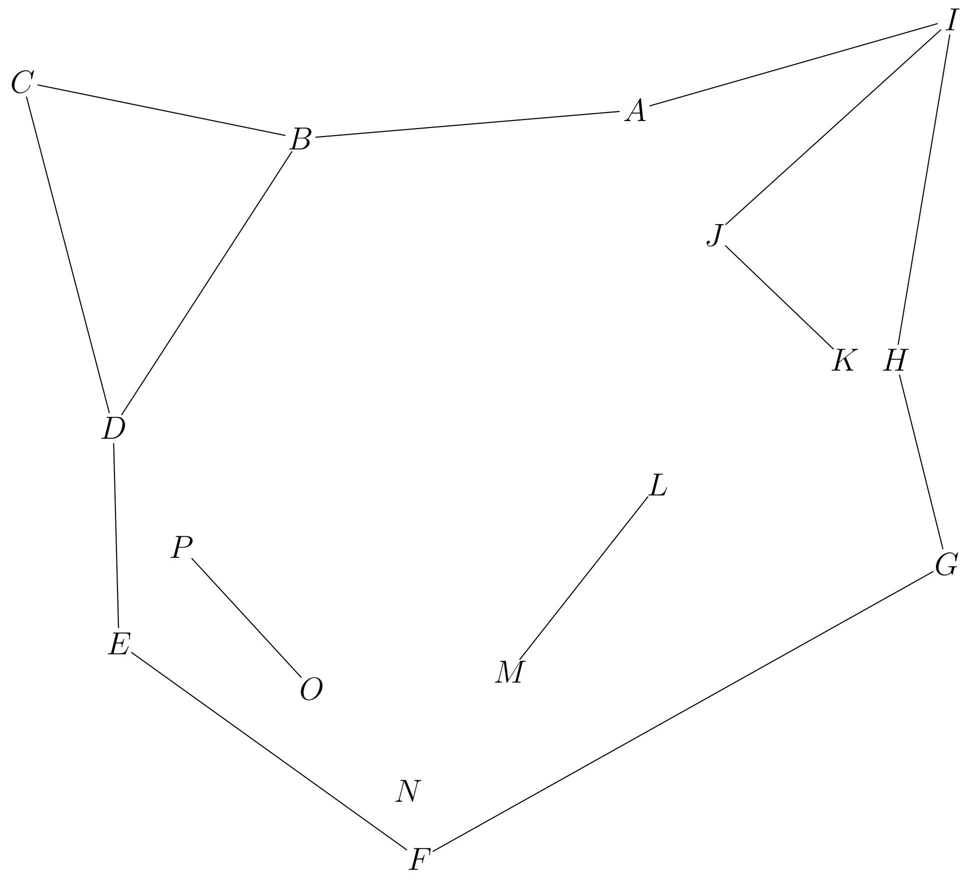
Lecture 13: Connectivity and Trees

Motivation: getting around! If graph is servers, how to route messages from one particular server to another? Road networks, driving directions. (Let's pretend for today that one-way roads don't exist!). What about efficiency? Shortest route, or fastest route (might not be the same!). In 6.1210 we'll study lots of optimization problems like this.

But let's ***walk*** before we run.

1 Walks

Definition 1. *A* walk *is a sequence of vertices* $(v_0, v_1, \ldots, v_k)$ *that follows edges:* $\{v_i, v_{i+1}\} \in E(G)$ *for each* i*. Can backtrack, revisit vertices or edges, anything goes! This walk has* length k*. (NB: there are* k edges*, and* $k+1$ vertices*.)*



Example: $(J, K, J, I, A, B, C, D, B)$

Definition 2. *A **trail** is a walk that does not repeat edges.*

Example: (K, J, I, A, B, C, D, B)

Definition 3. *A **path** is a trail that does not repeat vertices.*

Example: (K, J, I, A, B, C, D)

Definition 4. *In a graph G, vertices a and b are **connected** iff there exists a walk from a to b.*

Example: J and B are connected; A and L are not.

Q: Is P connected to itself? I.e., does there exist a walk from P to itself?

A1: Yes, (P, O, P) - walk of length 2. But what about N? Is N connected to itself?

A2: Still yes, the walk (N) has length 0, and starts and ends at N. Empty walks are fine!

Possible confusion: ***connected to*** is more general than ***adjacent to***; the latter means "share an edge".

Definition 5. *The whole graph G is **connected** iff every pair of vertices is connected. "Can get from anywhere to anywhere".*

Theorem 1. *If there exists a walk from a to b, then there exists a* path *from a to b.*

Proof. Common idea: take the longest or shortest! Since we know there exists at least one walk from a to b, pick a *shortest* walk, $(a = v_0, v_1, \ldots, v_k = b)$. If any vertex is used twice, say $v_i = v_j$ where $i < j$, we can skip over the middle: use $(v_0, \ldots, v_i)$, followed by $(v_{j+1}, \ldots, v_k)$. Since $v_i = v_j$, these stitch together to a walk from a to b, with length $k - (j - i) \leq k - 1$, which contradicts assumption that k was minimal. So the shortest walk from a to b is a path. □

2 Connected Components

From picture, seems like we should be able to split graph into smaller connected pieces. These are known as ***connected components***.

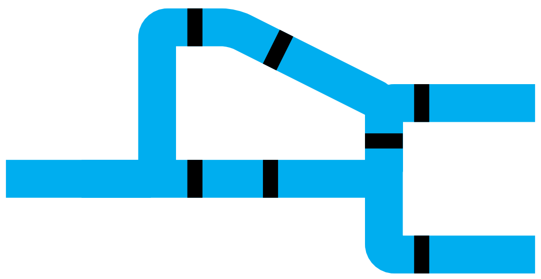
Useful! In google maps, there used to be an easter egg where asking for driving directions from USA to Japan would instruct you to find a Kayak. Since there are no roads connecting them, might as well consider the American road network as a completely separate graph from the Japanese road network. For many graph problems, if we just look at one connected component at a time, we can pretend that the graph is in fact connected!

Definition 6. *For $v \in V(G)$, the **connected component** of v is the subgraph* induced *by the set of vertices connected to v (aka reachable from v).*

Theorem 2. *Each connected component is, in fact, connected. Furthermore, these connected components form a partition of G: every vertex and every edge belongs to precisely one connected component.*

Proof. Idea: if v's and w's connected components have any vertices in common, then they are identical! □

3 Bridges of Königsberg



River Pregel ca. 1700, in the German city of Königsberg (now Kaliningrad, Russia).

Goal: find a walk that crosses each bridge exactly once and returns to starting point.
Note: Königs*berg* means King's *Mountain*, cf. burg (fortress) as in Hamburg. Königsberg has a fortress and a maximum elevation of about 25m.

Definition 7. *A walk is **closed** iff it begins and ends at the same vertex.*

Definition 8. *A **tour** is a closed trail.*

Definition 9. *A **cycle** is a tour that has positive length* and *does not repeat vertices (except at the very start/end).*

Definition 10. *A tour is **Eulerian** if it uses every edge exactly once and visits every vertex.*

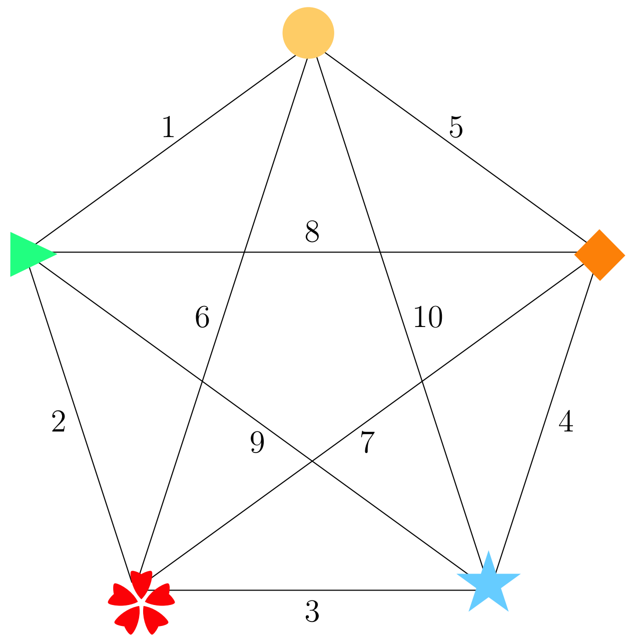
Definition 11. *A trail is **(semi-)Eulerian** if it uses every edge exactly once and visits every vertex.*

Definition 12. *A graph is **Eulerian** iff it has an Euler tour.*

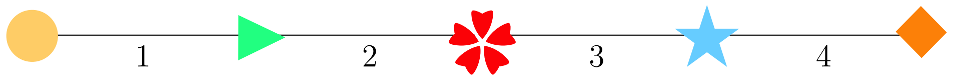
Definition 13. *A graph is **semi-Eulerian** iff it has an Euler trail.*

Warning: Be careful with terminology here!
In the context of trails, "semi-Eulerian" can be replaced with "Eulerian" or "Euler", and "trail" can be replaced with "walk" or "path".
In the context of tours, "Eulerian" can be replaced with "Euler", and "tour" can be replaced with "circuit" or "cycle".
Some of these are misnomers! An Euler cycle is not necessarily a cycle; an Eulerian path is not necessarily a path, nor does it make a graph Eulerian.

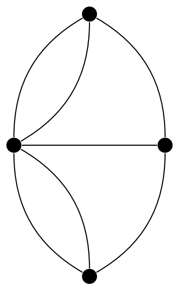
Example: K_5 is Eulerian (edges are numbered in order of Euler tour).



Example: A line graph is semi-Eulerian (edges are numbered in order of Euler trail).



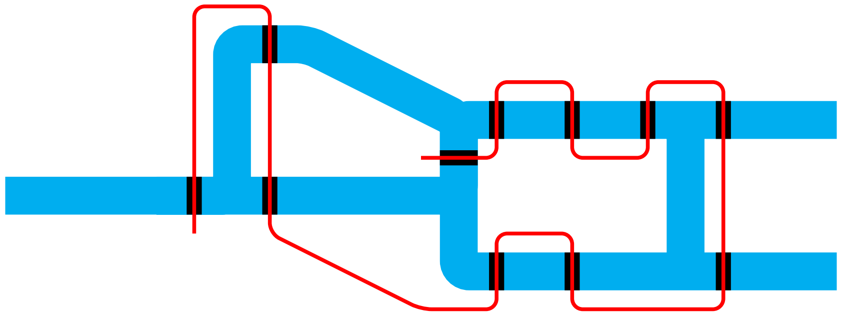
The Königsberg problem (solved by Euler in 1736) was basically the start of Graph Theory as a field. It is the question of whether the (multi)graph below is Eulerian.



Observations: if a graph G has an Euler Tour, then

- every vertex must have even degree. Why? enter and exit the same number of times!
- G must be connected. Why? Following the tour hits all vertices!

Immediately, Königsberg graph doesn't work, because there are odd vertices. In fact, they're *all* odd! However, it turns out that the bridges of *Kaliningrad* are now semi-Eulerian. The city was razed and rebuilt during/after WWII and now looks more like this (one Euler trail in red):



Are the observations sufficient? Do all connected graphs with all even degrees have Euler Tours? Yes!

Theorem 3. *A connected simple graph has an Euler tour if and only if all vertices have even degree.*

Proof. For the first direction, suppose G is a connected simple graph with an Euler tour W. Let v be a vertex of G. If we follow W from v, then for every time we leave v, we must return to v. Each of these departures and arrivals follows a distinct edge, and we have the same number of departures as arrivals, so v has even degree.

For the converse direction, let W be a maximum-length walk in G with no repeated edges. We'll show W is an Euler tour.

First, claim W is closed. If not, say its endpoints are $a \neq b$. Then a and b have an odd number of unused edges ("unused" means not belonging to W), and every other vertex has an even number of unused edges. This is because every time we enter-and-then-exit a vertex, we use two of its edges. So b has an odd number of unused edges, so it can take another step and make W longer. Contradiction, since we assumed W was already longest!

Now, claim that every edge in G is also in W. If not, say $\{u, v\}$ is unused. Let w be a vertex on W. Since G is connected, there is a walk π from w to u and then (immediately) to v. π contains an edge not in W, so let $\{s, t\}$ be the *first* edge in π that is not in W. Then s is on W, so since W is closed, we may start at s, and then follow W back to s and then to t. This gives a longer walk than W with no repeated edges. Contradiction, since we assumed W was already longest! □

Corollary 4. *A connected graph has an open Euler trail (doesn't start and end in the same place) iff exactly two vertices have odd degree.*

Proof. As before, one direction is easy. For the converse direction, suppose G is a connected graph, and exactly two vertices u and v have odd degree. We now have three cases:

- Case 1 (G does not contain the edge $\{u, v\}$): Create G' by adding edge $\{u, v\}$ to G. G' has an Euler tour, from which we may remove $\{u, v\}$ to get an Euler trail in G.
- Case 2 (G contains the edge $\{u, v\}$, and removing it does not disconnect G): Create G' by removing edge $\{u, v\}$ from G. G' has an Euler tour, to which we may add $\{u, v\}$ to get an Euler trail in G.

- Case 3 (G contains the edge $\{u, v\}$, and removing it disconnects G): Remove edge $\{u, v\}$ from G to create *two* connected graphs G' containing u and G'' containing v. Each of G', G'' has an Euler tour. Create an Euler trail for G by starting at u, following the Euler trail of G' back to u, taking edge $\{u, v\}$, and then following the Euler trail of G'' back to v.

These cases are exhaustive, and in each case G has an Euler trail. □

4 Trees

Trees are extremely useful. They come up all over computer science. Many data structures are based on trees, many algorithms make critical use of trees. They're some of the most basic important graphs you can learn about.

Lots of equivalent ways to define trees, each with their own insights.

The usual definition involves cycles; recall:

Definition 14. *A* ***cycle*** *is a closed walk* of length at least 3 *that doesn't repeat edges or vertices, except that the starting and ending vertices are the same.*

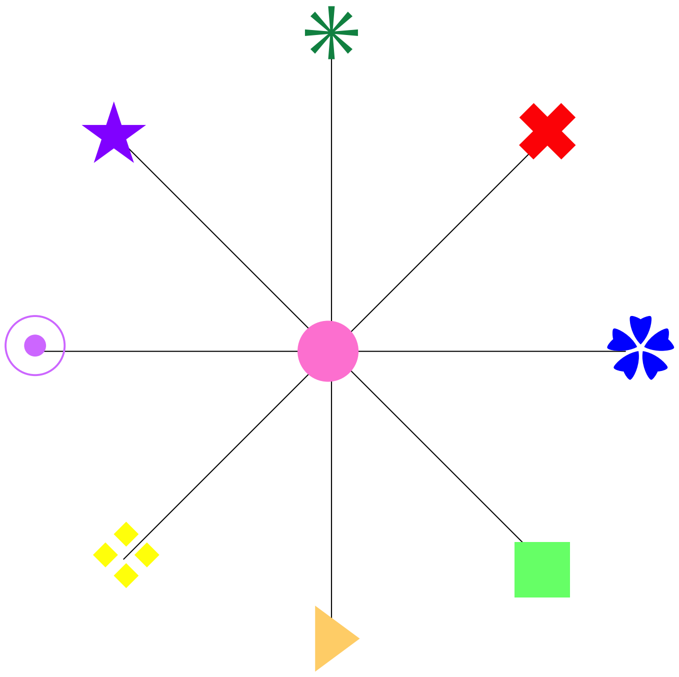
Non examples: empty walk doesn't count. Back-and-forth across a single edge doesn't count.

Idea: trees are minimal, stripped down, without redundancy. If roads are expensive to build, want to build as few as possible, while still being able to get from anywhere to anywhere.

Cycles are a form of "redundancy" (can go either way around the cycle), so trees should avoid cycles:

Definition 15. *A* ***tree*** *is a connected,* ***acyclic*** *graph. (Acyclic means no cycles.)*

Example:



A perhaps surprising fact:

Theorem 5. *A tree with n vertices has exactly $n-1$ edges.*

Before proving this theorem, let's see our most useful workhorse for proving things about trees.

Theorem 6. *Every tree with $n \geq 2$ vertices has at least 2 vertices with degree 1. These are called **leaves**.*

Proof. Take a longest path $v_0, v_1, \ldots, v_k$. There's at least one edge because connected and $n > 1$, so the endpoints v_0, v_k are different. Claim: v_0, v_k are leaves. If not, then v_0 must connect to another vertex other than v_1. If it is another v_i, we've found a cycle! If it is something other than a v_i, we can make the path longer! □

Now we can prove that every tree with n vertices has $n-1$ edges:

Proof. We'll use induction on n, where $P(n)$ is the statement "every tree with n vertices has exactly $n-1$ edges".

Base case: $n = 2$. The only connected graph with 2 vertices has 1 edge, so we're good.

Inductive step: Assume $n \geq 2$ and assume $P(n)$: "every tree with n vertices has exactly $n-1$ edges".

Must prove $P(n+1)$: "every tree with $n+1$ vertices has exactly n edges".

So suppose T is a tree with $n+1$ vertices; we must show it has n edges. (Note: we're unpacking things slowly, to make sure we really are proving what we need to, namely $P(n+1)$.

This is how to avoid buildup error. It's tempting to say "let T be a tree with n vertices", but that's not what $P(n+1)$ asks for.)

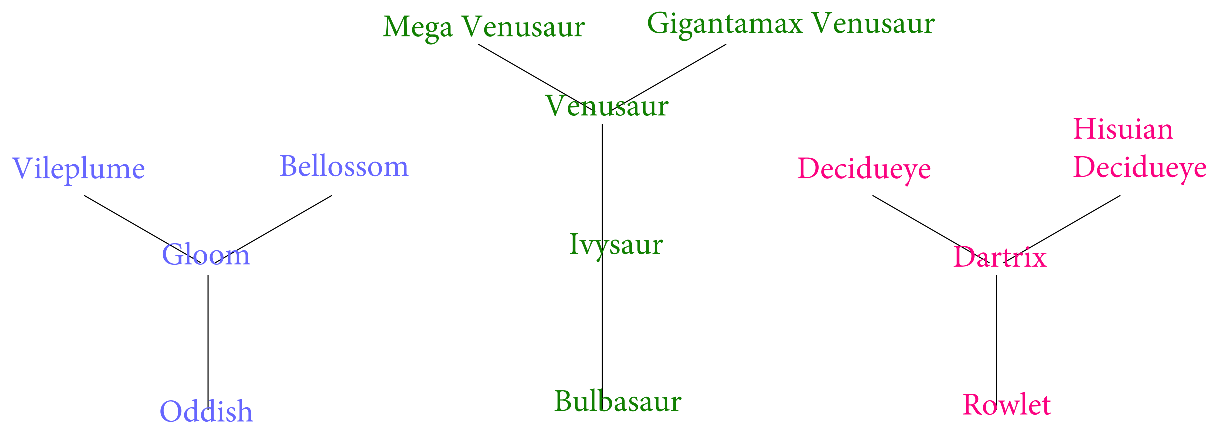
Let's prune the tree! By the lemma, T has at least one leaf, v. Let $T' = T - v$ be the graph that remains after removing the leaf v, along with its single edge. Claim T' is a tree with n vertices. (Will prove this in a bit; don't need to re-prove it every time!) By assumption $P(n)$, we know T' has $n-1$ edges. And T differs by just one edge (from the leaf), so T has n edges, as claimed.

But why is T' a tree? It is still acyclic: removing edges can't create new cycles. It is also connected: for two vertices u, w other than v, they are connected in T by a path, and this path can't use v because it has degree 1, so u, w remain connected to each other in T'. Acyclic and connected implies tree. $\square$

This is a very common pattern: remove a leaf, apply inductive hypothesis to smaller tree, then put the leaf back.

NB: Important that we removed a *leaf*, not some other arbitrary vertex! Removing any other vertex will not produce a connected graph, so what remains won't be a tree. (Show example in picture.) In general, acyclic graphs are known as *forests*, because each connected component is a tree.

Example:



With a bit more work (see book), can prove lots of different characterizations for trees:

Theorem 7. *If graph G has n vertices, the following are all equivalent to G being a tree:*

- *G is connected and acyclic. (This is the definition.)*
- *G is connected and has exactly $n-1$ edges.*
- *G is acyclic and has exactly $n-1$ edges.*
- *Every pair of vertices in G is connected by a* unique *path.*

- *G is minimally connected.*
- *G is maximally acyclic.*

See book for proofs.