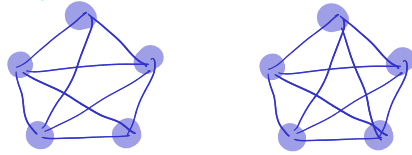
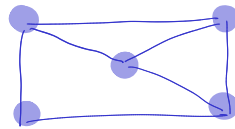


Can you draw these pictures, without ever crossing your path?

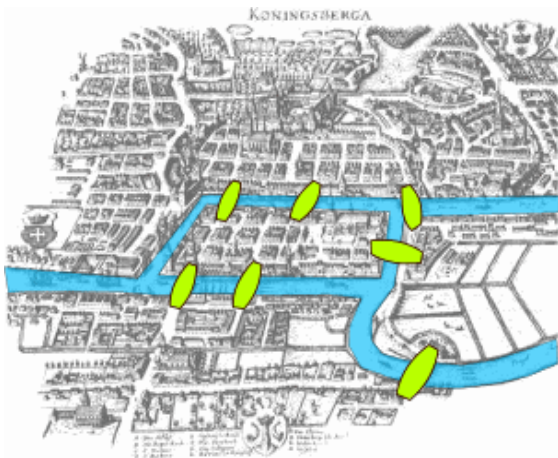


Can you draw this picture without ever lifting your pencil?



These are children problems, but also real-life problems in graph theory, namely to know whether a graph is planar, or similar to know if a graph is Eulerian.

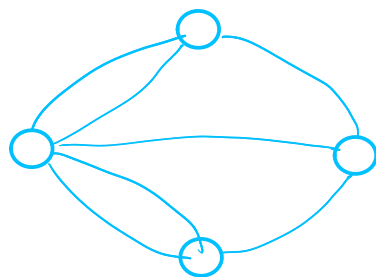
The first problem: Seven bridges of Königsberg (Euler, 1736)



Euler was wondering if one can go from one place in the Königsberg area, and back to that original place, by taking every bridge exactly once.

(This is considered to be the first solved problem in graph theory).

A modelisation of the problem:



This graph model the areas of the city. There is no need to know the exact location of each bridge.

Remarks:

- Since we have to go back where we started, we do not care where we start.
- Everytime we go from a location to another and back, we cross 2 bridges adjacent to that location.

Since every island has an odd number of bridges, it is not possible to visit all the islands by taking every bridge exactly once. ②


Some definitions

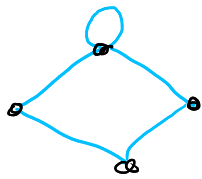
A graph G is made of a set of vertices (modeling some objects), and a set of relations between two vertices, called the edges. We denote $G = (V, E)$ for the graph with vertices V and edges E . Any edge is a pair of two vertices called the endpoints.

We draw a graph (on paper or on the computer) by representing the vertices as points, and we draw a curve between two vertices if they are endpoints of the same edge. We can draw differently the same graph.

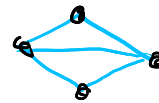
Example



A loop is an edge whose endpoints are the same vertex. 
Multiple edges are edges having the same pair of endpoints.
A simple graph is a graph having no loop nor multiple edges.



Not simple graphs



Simple graph

When uv (or equivalently) vu is an edge, we say the vertices u and v are adjacent, or that they are neighbors.

Subgraphs and containment

A graph $G' = (V', E')$ is a subgraph of $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. We then say that G' is contained in G , denoted $G' \subseteq G$.

Example

Every graph with n vertices is a subgraph of the complete graph with $m \geq n$ vertices.

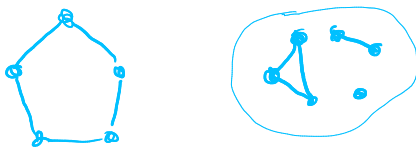
A graph is connected if, for every pair of vertices, there is a path (i.e. a sequence of edges) between them that belongs to the graph. It is otherwise disconnected.

Some important problems in graph theory

1. Acquaintances

Do every set of six people contain at least three mutual acquaintances or three mutual strangers?

That question can be represented using a graph. Every person is a vertex, and there is an edge between two persons if they know each other. Here, we assume knowing each other is a mutual relation, i.e. knowing a celebrity usually does not count.



As a homework, you will have to prove your solution to this statement.

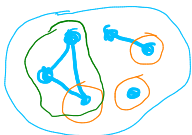
Two graphs. The first one is a 5-vertex graph with no three mutual strangers, nor three acquaintances.

The second one has six vertices, and contain both three mutual strangers and three acquaintances (a clique).

Some useful vocabulary:

A clique in a graph is a set of pairwise adjacent vertices, i.e. a complete subgraph.

An independent set is a subset of vertices with no adjacent pairs.



• A clique

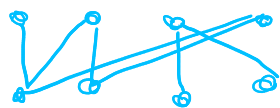
• An independent set

2. Job assignments

If there are m jobs and n people, not all qualified for all the jobs, is there a way we can fill all the jobs?

Definition

A bipartite graph is the disjoint union of two independent sets.



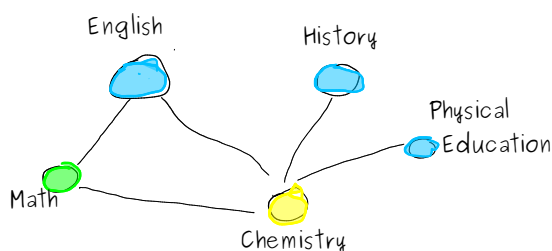
people
jobs

The edges are between a job and a qualified person for that job.

(The jobs cannot all be filled in this example).

3. Scheduling and avoiding conflicts

My high school used to have a very long exam sessions at the end of the year, and there were still some conflicts. I wish the administrators knew graph theory...



Vertices: Subjects

Edges: If someone takes both subjects, i.e. eventual scheduling conflicts.

A coloring of a graph is a partition of a set into independent sets. Scheduling with no conflicts is equivalent to coloring. If we want to use the minimum time, we should use as few colors as possible.

Schedule:

1. History-English-PE
2. Chemistry
3. Math

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Sections 1.1.1 and 1.1.2.

We saw last class that two graphs are the same if they are differently, as long as we are simply "moving the vertices". The goal of today's lecture is to make this statement more formal. One tool we will use is adjacency and incidence matrices. We will as well start classifying the graphs.

Matrices: adjacency matrix and incidence matrix

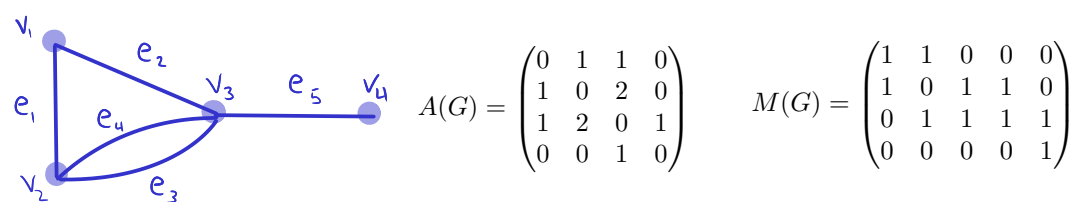
Let $G=(V, E)$ be a graph without any loop (it does not have to be a simple graph). We number the vertices from 1 to n and the edges from 1 to m .

The adjacency matrix of G , written $A(G)$, is the matrix whose (i,j) -entry is the number of edges with endpoints the vertices i and j .

The incidence matrix of G , written $M(G)$, is the n -by- m matrix whose (i,j) -entry is 1 if vertex i is an endpoint of edge j , and otherwise 0.

The adjacency matrix is always a symmetric matrix.

The graph on the left has the following adjacency and incidence matrices:



The degree of a vertex (in a loopless graph) is the number of edges incident to that vertex.

Isomorphisms

So when are two graphs the same? We will answer this question using the notion of a bijection. As a reminder, this is an injective and surjective function, or a one-to-one correspondence.

An isomorphism from a simple graph G to a simple graph H is a bijection $f:V(G) \rightarrow V(H)$ such that every edge uv of G is mapped to the edge $f(u)f(v)$ of H . We then say G and H are isomorphic, denoted $G \cong H$.

This is equivalent to asking that there exists a simultaneous permutation of the rows and columns of the adjacency matrix of G that would yield the adjacency matrix of H .

Example

The following graphs are isomorphic:



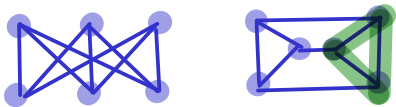
This is easily seen with the bijection that exchanges 1 and 3.

Remarks:

- Finding a bijection of the labels is the way to prove two graphs are isomorphic. However, to prove they are not isomorphic, there are many ways. For example, if the list of degrees is not the same, you will never be able to find an isomorphism. Or if the number of edges (or edges) do not correspond. Among others.
- The isomorphism relation is an equivalence relation, i.e. this is a symmetric relation ($G \cong H$ iff $H \cong G$), a transitive relation ($G \cong H$ and $H \cong J$ imply $G \cong J$) and a reflexive one ($G \cong G$). That means that we can split the graph into equivalence classes.

Example

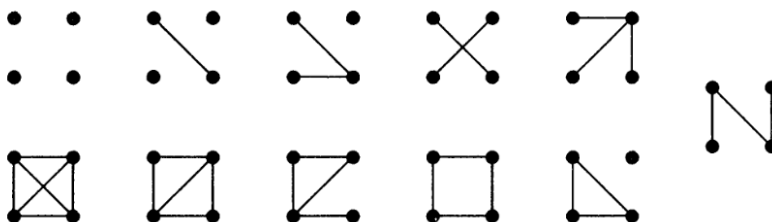
The following graphs are not isomorphic. They both have six vertices, all of degree 3, and nine edges, and they are both connected, but one is bipartite and the other is not. Since they don't have the same properties, they are not isomorphic.



No triangle appear in the first graph.

Example

All the isomorphism classes for graphs with 4 vertices are



Special graphs

There are some graphs that have special names, and that turns out to be handy for whenever we want to use them or to classify them.

Complete graphs: Graphs with n vertices and $\binom{n}{2}$ edges.

K_n

Example: K_5



Complete bipartite graphs: Bipartite graphs with independent sets of size s and r , with sr edges.

$K_{s,r}$

Example: $K_{4,2}$



Paths: Connected graphs, with all the vertices of degree 2, except at most two who have degree 1.

P_n

Example:

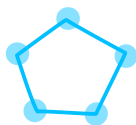
P_4



Cycles: Paths with as many edges as vertices.

C_n

Example: C_5



The complement of the graph G is the graph that has the same vertices and whose edges are all the edges that do not belong to G :
 $K_{|V|} - E(G) = \bar{G}$.

A graph G is self-complementary if its complement \bar{G} is isomorphic to G .

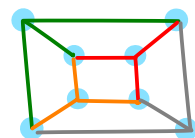
Example: C_5 is self-complementary.



A decomposition of a graph is a list of subgraphs in which every edge appears exactly once.

Example: The cube decomposed into copies of $K_{1,3}$

Note: $K_{1,3}$ is often called the claw.



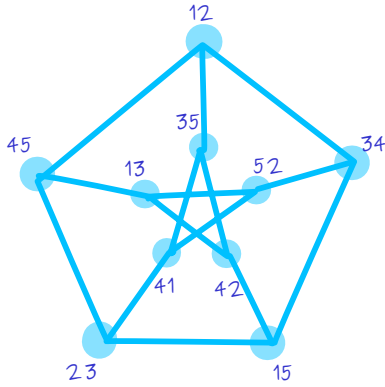
Proposition

A graph G is self-complementary if and only if the complete graph is a decomposition into two copies of G .

The Petersen graph

The Petersen graph is a 10-vertices graph with 15 edges that is very famous, as it is an example or a counter-example to many phenomena.

The Petersen graph is the graph of 2-element subsets of $\{1,2,3,4,5\}$, and there is an edge between 2 subsets if their intersection is empty.



Some properties of the Petersen graph:

- Two non-adjacent vertices share exactly one neighbor.
- The graph has no triangle, but is not bipartite.
- The shortest cycle in the Petersen graph has length 5. (The length of the shortest cycle in a graph is called the girth of the graph.)

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 1.1

Today's lecture aims to define the proper vocabulary to talk about trajectories and connectedness in graphs.

Definitions

Recall that a path is a graph whose vertices can be ordered without repetition (except maybe for the endpoints) in a sequence such that two consecutive vertices are adjacent. A path is a u,v -path if it starts at vertex u and ends at vertex v .

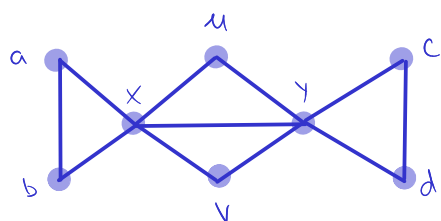
A walk is a list $(v_0, e_1, v_1, \dots, e_k, v_k)$ of vertices and edges such that the edge e_i has endpoints v_{i-1} and v_i . A walk is a u,v -walk if its endpoints (the first and last vertices of the walk) are u and v . If there is no multiple edges, we can write the walk as (v_0, v_1, \dots, v_k) .

A trail is a walk with no repeated edge. Similarly, a u,v -trail has endpoints u and v .

The points that are not endpoints are internal vertices.

The length of a walk, trail, path or cycle is its number of edges. A walk or a trail is closed if its endpoints are the same.

Example



$(a, x, a, b, x, u, y, x, a)$ specifies a closed walk, but not a trail (ax is used more than once).

(a, b, x, u, y, x, a) specifies a closed trail.

The graph contains the five cycles (a, b, x, a) , (u, y, x, u) , (v, y, x, v) , (x, u, y, v, x) and (y, c, d, y) .

The trail (x, u, y, c, d, y, v, x) is not an example of a cycle, since vertex y is repeated (so it is not a path).

Lemma

2

Every u,v -walk contains a u,v -path.

Proof

The proof can be done using the principle of strong induction, and we induce on the number of edges.

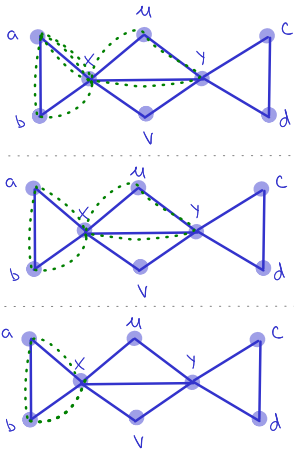
Base case: No edge, $u=v$ is the only vertex in the graph. Only walk has length 0, and is therefore a path.

Induction hypothesis: Assume that, for a walk with $k < n$ edges, there is always a path with the same endpoints.

Induction step: The walk has n edges. There are two cases: either there is no repeated vertex or only the endpoint is repeated, and then the walk is already a path, or there is a repeated vertex x . In the latter case, we delete the edges between the first and last occurrences of x , which leaves us with only one copy of x , and a u,v -walk with fewer than n edges. We can thus use the induction hypothesis to conclude that there exists a u,v -path in the u,v -walk.



Example: The u,v -walk from previous page.



In the walk $(a, x, a, b, x, u, y, x, a)$, we delete what happens between the first two occurrences of a , and get the closed walk (a, b, x, u, y, x, a) . Then we delete what happens between the two occurrences of x , and get the cycle (a, b, x, a) , which is a path.

Connectedness, components and cuts

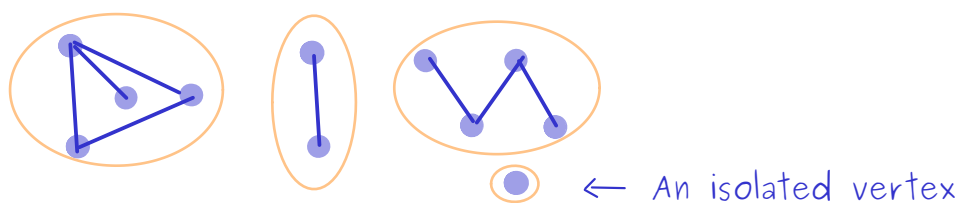
Recall that a graph is connected if and only if there exists a path between u and v for every pair of vertices $\{u, v\}$.

A component of a graph G is a maximal connected subgraph.

A component is trivial if it has no edges; in this case, the unique vertex is said to be an isolated vertex.

Example

The following graph has 4 components, each of which are circled in orange.



Proposition

Every graph with n vertices and k edges has at least $n-k$ components.

Proof

The proof can be done by induction on k . The case of $k > n$ is obvious, since the number of components is always nonnegative.

Base case: If $k=0$, then each of the n vertices are isolated, and there are n components.

Induction hypothesis: Assume that a graph with $k-1$ edges and n vertices has at least $n-k+1$ components.

Induction step: Let $G=(V,E)$ with $|V|=n$ and $|E|=k$. Remove the edge e to get $G-e$. The component of G containing e can either be split into two components by removing e , or stay a component. So G has either the same number of components as $G-e$, or one fewer. By induction hypothesis, $G-e$ has at least $n-k+1$ components, so G has at least $n-k$.

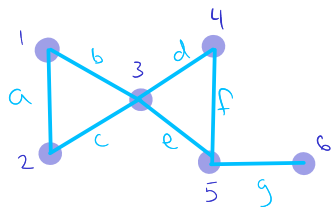


In the last proof, we had to distinguish the cases where removing the edge was creating a new component or not. An edge whose deletion creates new component has a special name:

A cut-edge or cut-vertex of a graph is an edge or vertex whose deletion increases the number of components. We write $G-e$ or $G-M$ for the subgraph of G obtained by deleting an edge e or a set of edges M ; we write $G-v$ and $G-S$ for the graph obtained by deleting a vertex v or a set of vertices S along with their incident edges.

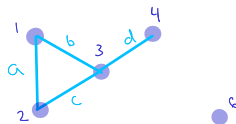
A subgraph obtained by deleting a subset of vertices and their incident edges is an induced subgraph: we denote it $G[T]$ if $T=V \setminus S$ and we deleted the vertices in S .

Example



Vertices 3 and 5 are cut-vertices, and the edge g is the only cut-edge.

The induced subgraph for the vertices 1, 2, 3, 4 and 6:



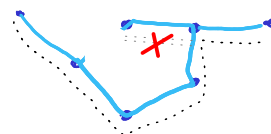
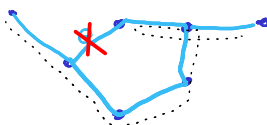
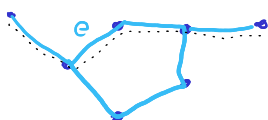
Theorem

An edge is a cut-edge if and only if it belongs to no cycle.

Proof

Let $e=uv$ be an edge in the graph G , and let H be the component containing e . We can restrict the proof to H , since deleting e does not influence the other components. We want to prove that $H-e$ is connected if and only if e is in a cycle in H .

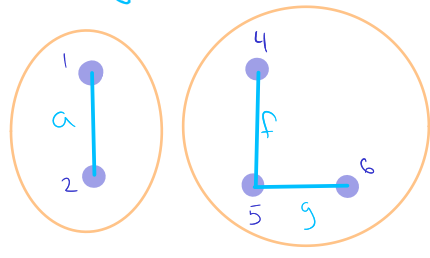
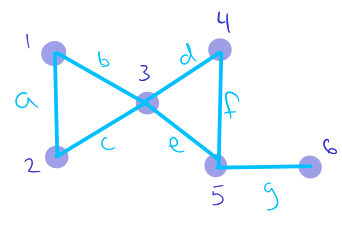
If e is in a cycle c , $c-e$ is a path P between v and u avoiding the edge e . To show that $H-e$ is still connected, we need to show that, for every pair of vertices $\{x, y\}$, there is a path between x and y . Since H is connected, there exists in H such a path. If that path does not contain e , it is still in $H-e$. Otherwise, replace e by P , and remove an edge from that path everytime it appears twice consecutively.



If $H-e$ is connected, then there exists in it a path P between u and v . Hence, adding edge $e=uv$ creates the cycle $P+e$. ■

The last theorem allows us to characterize cut-edges. Would such a theorem be possible for cut-vertices? The following example proves that asking for it to be outside a cycle is not a requirement for a cut-vertex, since vertex 3 is a cut-vertex, and belongs to two cycles:

Removing vertex 3:



Two connected components

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 1.2

Today's lecture aims to give the important properties of bipartite graphs. We will also define Eulerian circuits and Eulerian graphs: this will be a generalization of the Königsberg bridges problem.

Characterization of bipartite graphs

The goal of this part is to give an easy test to determine if a graph is bipartite using the notion of cycles: König theorem says that a graph is bipartite if and only if it has no odd cycle.

Lemma

Every closed walk of odd length contains an odd cycle. This is called an odd closed walk.

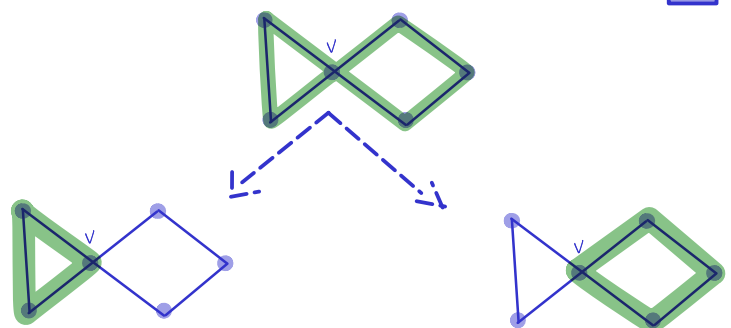
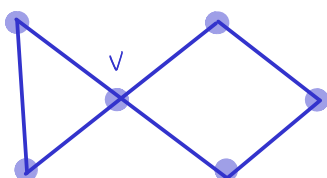
Proof

We prove it using strong induction on the length of the walk (i.e. the number of edges).

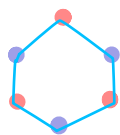
Base case: length 1. The walk is a loop, which is an odd cycle.

Induction hypothesis: If a walk has odd length at most n , then it contains an odd cycle.

Induction step: Consider a closed walk of odd length $n+1$. If it has no repeated vertex (except the first and last one), this is a cycle of odd length. Otherwise, assume vertex v is repeated. We can split the walk into two closed walks starting and ending at v , one of even length, and one of odd length smaller than n . By induction hypothesis, the latter contains an odd cycle.



That lemma will be helpful for characterizing bipartite graphs. Of course, bipartite graphs can have even cycles, which starts in one independent set and ends there.



We can represent the independent sets using colors.

Theorem (König, 1936)

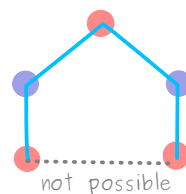
A graph is bipartite if and only if it has no odd cycle.

Proof

Notice that a graph is bipartite if and only if all its components are bipartite. So we do the proof on the components.

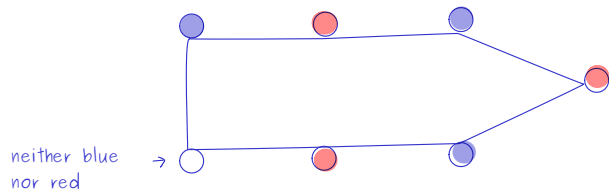
⇒ We prove the contrapositive: if it has an odd cycle, it is not bipartite.

Since every cycle must end at the vertex where it starts, it starts and ends in the same independent set. Since every edge is going from one set to the other, we alternate between the two sets. At the end of the cycle, we cannot close it, since we would need to change the set of the first vertex. Hence, if a connected graph is bipartite, it has no odd cycle.

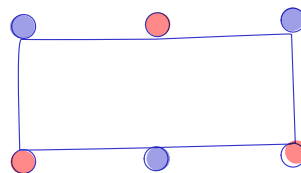


⇐ We still need to prove that a connected graph without odd cycle is bipartite. If the graph has only one vertex, it is bipartite.

Otherwise, start at vertex u , and color its neighbors with color blue. Then, color the neighbors of the blue vertices in red, and repeat this process by coloring the neighbors of the red vertices in blue, until all vertices have been colored. I claim that no vertex will change color in that process; assume otherwise, that v is changing color. That would mean that there exists a path of odd length from u to v (the one that colors v in blue), and a path of even length doing it (the one that colors v in red). The combination of these two paths is an odd walk, and contains an odd cycle, which is prohibited by the hypothesis. Hence, the coloring is well defined, and the two colors represent independent sets. The graph is bipartite. ■



odd cycle



even cycle

Technique for checking whenever a graph is bipartite:

- If it is bipartite, prove it by finding two independent sets.
- If it is not bipartite, find an odd cycle.

Eulerian circuits

A graph is Eulerian if it has a closed trail containing all the edges.

The graph in the Königsberg bridges problem is not Eulerian. We saw that the fact that some vertices had odd degree was a problem, since we could never return to that vertex after leaving it for the last time.

Theorem

A graph is Eulerian if and only if it has at most one nontrivial component (i.e. component with edges), and if every vertex has even degree.

Proof

We first prove \Rightarrow by proving the contrapositive: if a graph has more than one non-trivial component, or if there is a vertex of odd degree, then the graph is not Eulerian.

If a graph has at least two non-trivial components, there can't be a walk going through all the edges, since they are in separate components.

If a graph has a vertex of odd degree, we are in the case of the Königsberg bridges: we can leave the vertex more often than we can come back (or vice-versa), and thus our trail cannot be closed.

\Leftarrow We need to prove that a connected graph with only vertices of even degrees is Eulerian. We can ignore the isolated vertices for this since we are focusing on the edges. The following lemma is useful:

Lemma

If every vertex of a graph has degree at least 2, then it contains a cycle.

Proof

Let P be a maximal path in that graph. If it is a cycle, we are done. Otherwise, let u be an endpoint of P .

Since it has degree at least 2, u has a neighbor v not in P . But since P is maximal, that means that v is already in P , and the edge uv completes the cycle.

Proof of the theorem (continued)

We proceed by induction on the number of edges.

Base case: 0 edge, the graph is Eulerian.

Induction hypothesis: A graph with at most n edges is Eulerian.

Induction step: If all vertices have degree 2, the graph is a cycle (by definition) and it is Eulerian. Otherwise, let G' be the graph obtained by deleting a cycle. The lemma we just proved shows it is always possible to delete a cycle. By induction hypothesis, G' is Eulerian. To build an Eulerian circuit in G , start by the cycle we just deleted, and append the Eulerian circuit of G' .

Proposition

Every graph with only vertices of even degree decomposes into cycles.

Eulerian circuits are closed trails that pass through all edges. A similar property is being Hamiltonian: a Hamiltonian circuit is a circuit that passes through all vertices exactly once. A Hamiltonian graph is a graph with a Hamiltonian circuit.

Today, we are doing a bit of combinatorics and will deduce some properties on the degrees, number of edges and number of vertices.

We already defined the degree of a vertex in a loopless graph to be the number of edges incident to it.

For a general graph, define the degree $d_G(v)$ of the vertex v to be the number of edges incident to it, with each loop counted twice.

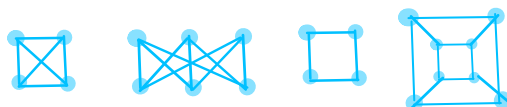
The maximum degree of a vertex is denoted $\Delta(G)$ and the minimum degree is denoted $\delta(G)$.

A graph is said to be regular if $\delta(G) = \Delta(G)$.

The order of a graph $G = (V, E)$ is $|V|$, as the size of G is $|E|$.

Example

- K_n is a regular graph. Each vertex has degree $n-1$.
- $K_{m,n}$ is regular if and only if $m=n$. Then, the degree is always n .
- A connected regular graph that has the same order and size is a cycle.
- Hypercubes are regular graphs.



Counting and bijections

Proposition (degree-sum formula)

If $G = (V, E)$ is a graph, then $\sum_{v \in V} d_G(v) = 2|E|$.

Proof

For each edge, there are two endpoints (maybe equal). If the endpoints are different, this edge contributes for 1 in the degree of two different vertices. If the edge is a loop, it adds 2 to the degree of the vertex it is incident to. So either way, every edge accounts for 2 in the total degree count. ■

Corollary

In any graph $G = (V, E)$, the average degree is $2|E|/|V|$, and $\delta(G) \leq 2|E|/|V| \leq \Delta(G)$.

Corollary

(2)

Every graph has an even number of vertices of odd degree.

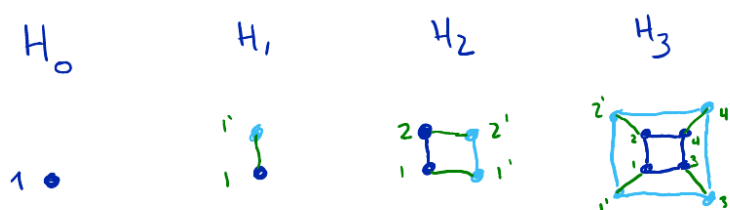
Corollary

A k -regular graph (i.e. a regular graph in which the degree of each vertex is k) has $k|V|/2$ edges.

Example: Hypercubes

The n -dimensional hypercube H_n is defined recursively as:

- H_0 is the simple graph with one vertex
- H_{n+1} is obtained by creating two copies of H_n and appending an edge between corresponding vertices in the two copies.



Proposition

H_n is regular, as each vertex has degree n .

Proof

The proof can be done by regular induction.

The base case is H_0 , and it has no edge.

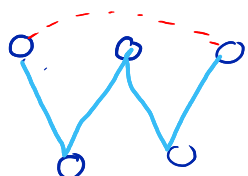
Induction hypothesis: The n -dimensional hypercube H_n is n -regular.

Induction step: The $(n+1)$ -dimensional hypercube is made of two copies of H_n , and we add an edge between every pair of similar vertices in the two copies. This way, we add exactly one to the degree of each vertex from H_n , and that degree is, by induction hypothesis n .



Proposition

If $k > 0$, then a k -regular bipartite graph has the same number of vertices in its two independent sets.



Either not bipartite or not regular.

Proof

Since the graph is regular, all vertices have degree k . If there are m edges in total, the sum of the degrees for all the vertices in one independent set is m , as every edge has exactly one endpoint in that set. Since the graph is k -regular, there are m/k vertices in each set, so the order of both sets is the same.

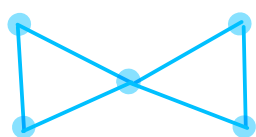


Vertex-deletion and reconstruction conjecture

Is it possible to reconstruct a graph if you have only a list of its subgraphs? There is a long-standing, and still open conjecture saying it is, and so far we know it is almost always possible (that being understood in a probabilistic sense).

For a graph G , a vertex-deleted subgraph is an induced subgraph $G-v$ obtained by deleting a single vertex v .

Example



has vertex-deleted subgraphs 4 x



1 x



Proposition

For a simple graph $G=(V,E)$ of order $n \geq 2$ and size m ,

$$m = \frac{\sum_{v \in V} \#E(G-v)}{n-2}$$

where $\#E(G-v)$ is the number of edges in the graph $G-v$.

Proof

We start with the summation, and we will prove the summation is equal to $m(n-2)$:

$$\sum_{v \in V} \#E(G-v) = \sum_{v \in V} |E| - d_G(v) = \sum_{v \in V} |E| - \sum_{v \in V} d_G(v) = mn - 2m$$

Conjecture (Reconstruction Conjecture – Kelly, Ulam, 1942)

If G is a simple graph with at least three vertices, then G is uniquely determined by the list of its vertex-deleted subgraphs (up to isomorphism).

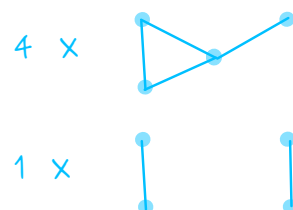
Note that the hypothesis that G has at least three vertices is important. Otherwise, we would find a counterexample with two vertices, since both simple graphs with two vertices have the same set of vertex-deleted subgraphs.



Example



has vertex-deleted subgraphs



To reconstruct the graph, we know that 4 vertices have degree $\#E(G)-4$ and 2 has degree $\#E(G)-2$. Using the proposition, the number of edges in G is $(2+4 \times 4)/3=6$. So the list of degrees is $(2,2,2,2,4)$, and the graph is connected.

That means that the vertices are in two cycles. The length of the cycles can be found by looking at the subgraphs: there is at least one cycle of length 3. Since the graph is simple, both cycles have length 3 and the graph has to be isomorphic to the bowtie.

Even though the conjecture is not proven, there are a number of cases that are known. Also, we can know some properties from the list of subgraphs; for example if the graph is connected.

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 1.3

Extremal problems consider the minimum and maximum numbers some statistics on a class of graphs can reach. We introduce some of the types of proofs useful in graph theory: Algorithmic, and by construction.

First example

In any simple graph (V, E) , the maximum number of edges is

$$\binom{|V|}{2} = \frac{|V|(|V|-1)}{2}$$

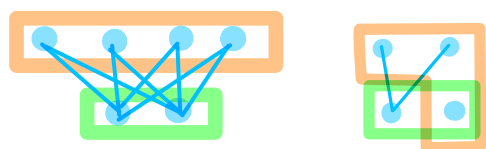
Proof

In a simple graph, there can be at most one edge per pair of distinct vertices. The maximum number of edges appear in $K_{|V|}$.

This is an extremal problem, since we are looking at the maximum number of edges. The class of graphs here is all simple graphs.

Example

In a bipartite graph with independent sets of size k and m , there can be at most km edges.



Independent sets of size 2 and 4, 8 edges at maximum. km is the number of edges of $K_{m,k}$.

Edges in connected graph

Proposition

The minimum number of edges in a connected graph with n vertices is $n-1$.

Proof

We need to prove two things:

- If a graph with n vertices has fewer than $n-1$ edges, it is not connected.
- There exists a connected graph with n vertices and $n-1$ edges.

(2)

Recall from last week (Friday), that a graph with n vertices and m edges has at least $n-m$ components. Hence, if $m < n-1$, the graph has at least 2 components and is not connected.

Also, the path with n vertices has $n-1$ edges and is connected, proving that the minimum is realized.



Remark (on the proof technique)

When giving the solution to an extremal problem, there are two parts to be proven:

- That the value we give is minimal (or maximal), i.e. that you cannot give a lower (respectively, higher) value.
- That this value can be realized on at least one graph of the class we consider.

Proposition

Let G be a simple graph with n vertices. If the minimum degree is $\delta(G) \geq (n-1)/2$, G is connected.

Proof

The minimum degree of the graph means that every vertex should have at least this number of neighbors, in a simple graph.

To prove that G is connected, we must show that there is a path between any pair of vertices $\{u, v\}$. We will in fact prove that there exists a path of length at most 2.

- If $\{u, v\}$ are adjacent, they are obviously in the same component.
- Otherwise, they share at least one neighbor w : There are $n-2$ other vertices, and the sum of their degree is $d(u) + d(v) \geq n-1$. Hence, $u-w-v$ is a path connecting them.



A bound is said to be sharp if improving it (reducing a lower bound or increasing an upper bound) would make the statement wrong.

The bound in the last problem is sharp. To prove it, we give an example of a graph with n vertices and minimum degree $\lfloor \frac{n}{2} \rfloor - 1$ that is not connected: This graph is the disjoint union of $K_{\lfloor \frac{n}{2} \rfloor}$ and $K_{\lfloor \frac{n}{2} \rfloor}$.



K_5 , degree 4



K_6 , degree 5

11 vertices

Minimum degree is 4, just under $5 = (11-1)/2$.

Graph is disconnected.

Bipartite subgraph

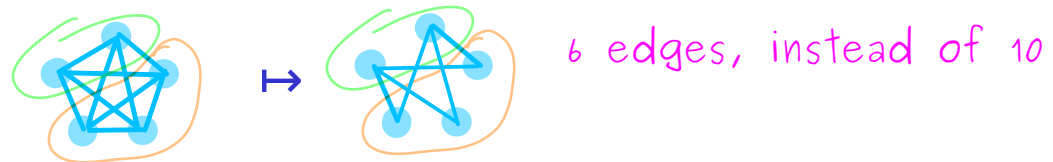
Here we prove that, given a graph G , we can always find a bipartite subgraph with at least a fixed number of edges. We give an algorithmic proof to construct the graph, but a proof can also be done by induction.

Theorem

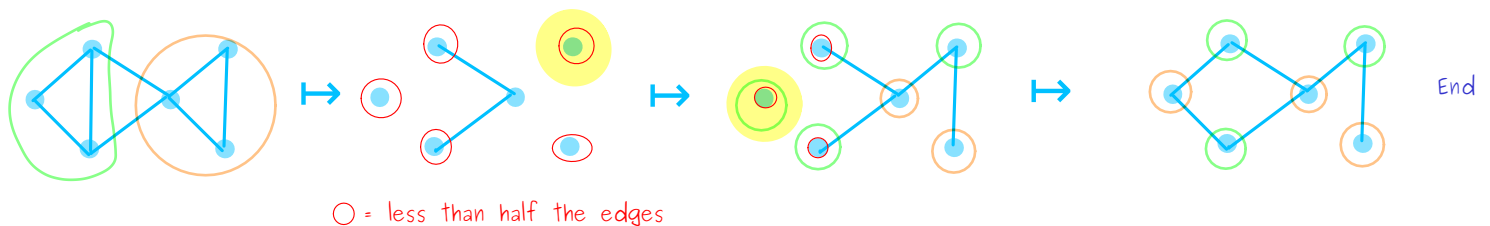
Every loopless graph $G=(V,E)$ has a bipartite subgraph with at least $|E|/2$ edges.

Proof (algorithmic)

We start with any partition of the vertices into two sets X and Y . Let H be the subgraph containing all the vertices, but only the edges with one endpoint in X and one in Y .



Let v be a vertex in X . If H has fewer than half the edges incident to v , then it means that v has (in G) more neighbors in X than in Y . To increase the number of edges in H , switch v to Y . The number of edges just increased.

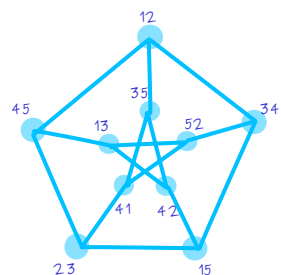


As long as H does not have at least half the edges of G at every vertex, there are vertices that can be swapped from X to Y or Y to X ; repeat this process. When it terminates, the number of edges in H is always at least half the number of edges of G . ■

Triangle-free graphs

A graph is said to be triangle-free if it has no three vertices that are all adjacent. In general, a graph G is H -free if it does not contain H as a subgraph.

The Petersen graph is triangle-free (but not bipartite).



Theorem (Mantel, 1907)

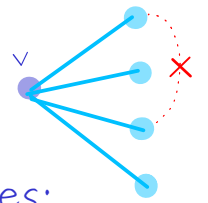
The maximum number of edges in a simple triangle-free graph with n vertices is $\lfloor \frac{n^2}{4} \rfloor$.

Proof

For the proof, we again need to prove two things:

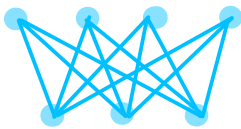
- that a triangle-free graph with n vertices cannot have more than $\lfloor \frac{n^2}{4} \rfloor$ edges.
- that there exists, for any n , a graph with n vertices and $\lfloor \frac{n^2}{4} \rfloor$ edges that has no triangle.

For the first part, assume the graph is triangle-free. Take a vertex v of maximal degree Δ . Its Δ neighbors cannot have edges among them. So every edge of G must have at least one endpoint in a non-neighbor of v , or in v itself. There are $n - \Delta$ such vertices. Each such vertex has degree at most Δ .



Therefore, we give an upper bound on the number of edges: the number of edges is at most $\Delta(n - \Delta)$ (because $n - \Delta$ is the number of vertices not adjacent to v). Maximizing $\Delta(n - \Delta)$ gives $\Delta = n/2$. Hence, the number of edges is at most $\lfloor \frac{n^2}{4} \rfloor$.

For the second part, we must prove that a triangle-free graph has $\lfloor \frac{n^2}{4} \rfloor$ edges. This is the case of $K_{\lceil \frac{n}{2} \rceil, \lfloor \frac{n}{2} \rfloor}$.



We can split 7 vertices into two sets of 3 and 4 vertices, which leads to 12 edges, which is the smallest integer below $49/4$.

We look at the list of degrees to get some information on the graph. We also look at what list of nonnegative integers can be the degree sequence of some graph.

Let G be a graph with vertices v_1, v_2, \dots, v_n . The degree sequence of G is the list $d(v_1), d(v_2), \dots, d(v_n)$. Usually, we write this sequence in decreasing order (and reorder the labels accordingly):

$$d_1 \geq d_2 \geq \dots \geq d_n \geq 0$$

Proposition

The nonnegative integers d_1, d_2, \dots, d_n are the degree sequence of some graph if and only if their sum is even.

Proof

We need to prove that the condition is both necessary and sufficient.

\Rightarrow (the condition is necessary) We already showed (last week) that the sum of the degrees in a graph is always even.

\Leftarrow (the condition is sufficient) This part of the proof is done by constructing a graph with a given degree sequence.

First, we consider all the vertices with odd degree (there is an even number of them). We pair them by drawing exactly one edge at each of these odd vertices. After this step, the number of endpoints to be added to every vertex is even, so we can add half this number of loops, making it a degree sequence.



Example

$(5, 3, 2, 1, 1)$ can be realized on a (non-simple) graph in this way:



Of course, this technique does not work for simple graphs, because of the loops. Moreover, 5 cannot be the degree of a vertex in a simple graph with 5 vertices.

A graphic sequence is a list of nonnegative integers that is the degree sequence of some simple graph. A simple graph with degree sequence d realizes d .

Characterization of graphic sequences

We already noticed the two obvious conditions for a nonnegative integers sequence to be graphic, i.e. the sum of degrees must be even and the maximal number cannot be greater than $n-1$. However, this is not enough, as shown with the degree sequence $(2,0,0)$, which must necessarily involve a loop.



Theorem (Havel 1955, Hakimi 1962)

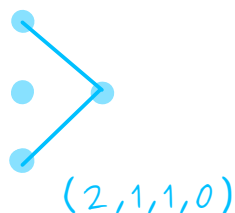
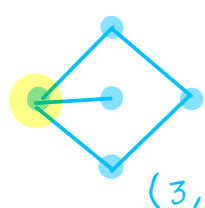
The only one-element graphic sequence is (0) .

For $n > 1$, an integer list d of length n is graphic if and only if d' is graphic, where d' is obtained by deleting its largest element (Δ) and 1 from the Δ next largest degrees.

Example

The graph below has degree sequence $d = (3, 2, 2, 2, 1)$.

It is obviously graphic by the picture. Here, $\Delta = 3$, and we obtain d' as $(1, 1, 1, 1)$. Notice that it is not the degree we obtain by deleting the highest-degree vertex (shown on the right), which would be $(2, 1, 1, 0)$. And $(1, 1, 1, 1)$ is also realizable, as shown below.



Proof (of theorem)

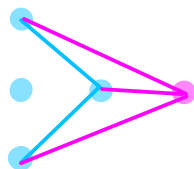
The case where there is only one vertex is obvious.

We need to prove that this condition is necessary and sufficient when $n > 1$.

\Leftarrow (sufficient) If d' is realizable, there exists a graph G' with vertices having d' as degrees. I want to add a vertex that has degree Δ greater than the largest degree of G' . To do so, I add the vertex and connect it to the Δ vertices with larger degrees in G' , realizing d .

$$d' = (2, 1, 1, 0)$$

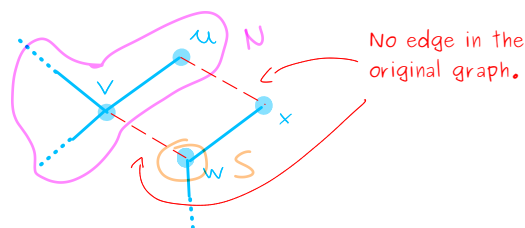
$$d = (3, 3, 2, 2, 0)$$



=> (necessary) There are two cases to consider. 1) The vertex v of degree Δ has neighbors that have the Δ next highest degree. Deleting v and its incident edges yield a graph with degree sequence d' .
 2) Consider the neighborhood of v (the vertex of higher degree) and call it N . Let S be the set of the Δ vertices having the highest degrees (except for v). Case 1) is when $N=S$, so here they are distinct. We will transform G to get $N=S$.

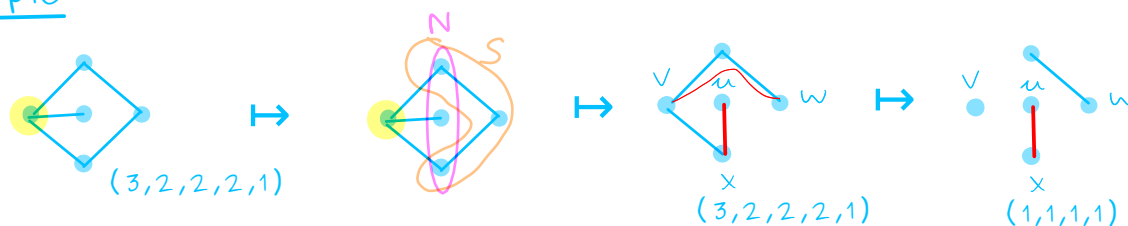
Take a vertex u in $N \setminus S$, so u is adjacent to v , but has a low degree, and take w in $S \setminus N$ (not adjacent to v , but high degree).

Since w has higher degree than u in $G \setminus \{v\}$, w has at least one neighbor x that is not adjacent to u .



By switching the edges uv and xw to vw and ux (from the blue to the red in the picture), we increase $|N \cap S|$. We repeat this process as long as $N \neq S$. When $N=S$, we use the first case.

Example



The case of loopless graphs

Multigraphs (even loopless) have a much easier characterization for degree sequences, as given by this theorem of Hakimi.

Theorem (Hakimi, 1962)

A sequence of decreasing nonnegative integers d_1, d_2, \dots, d_n is the degree sequence of a loopless graph if and only if its sum is even and

$$d_1 \leq d_2 + \dots + d_n$$

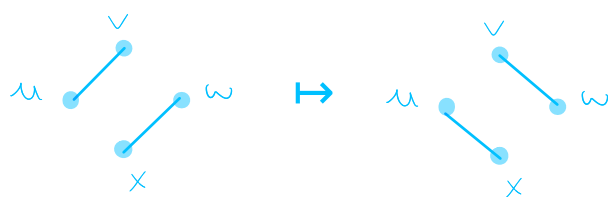
Proof is left as homework for next week's set.

Hint: You can proceed by construction, but it might be easier to do induction (not necessarily on the number of vertices).

Graphs with same graphic sequence

In the last proof, we exchanged the endpoints of some edges to get a new graph with the same graphic sequence.

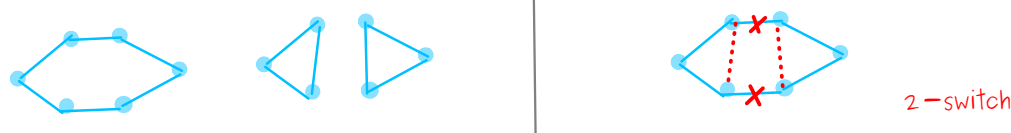
A 2-switch is the replacement of a pair of edges $\{uv, wx\}$ by $\{ux, vw\}$, provided ux and vw did not already exist in the graph.



Remark

A 2-switch always preserves the degree of each vertex.

Example



Both graphs have degree sequence $(2, 2, 2, 2, 2, 2)$.

Theorem (Berge 1973)

Two simple graphs G and H have the same graphic sequence if and only if there is a sequence of 2-switches from G to H .

The proof is omitted, but can be found on page 47 of the textbook. The condition is clearly sufficient, as the 2-switches preserve the degree of each vertex.

We introduce directed graphs and their terminology. Applications include Markov chains, automata and De Bruijn graphs.

A directed graph or digraph is made of two sets: the vertices, and a set of edges defined as ordered pairs of two vertices: a tail and a head. For one edge, the tail and the head are both endpoints, and we say the edge is from its tail to its head. We sometimes use the word arrow for the edges of a directed graph.

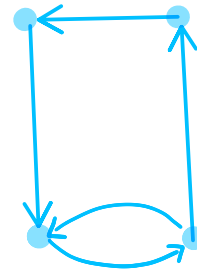
$u \rightarrow v$ Edge from u to v

Some examples

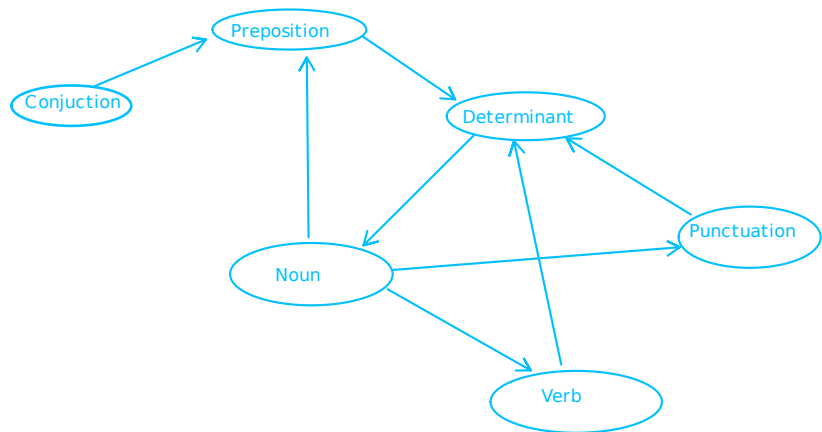
Trail etiquette



Car traffic around the Green



Parts of speech in the sentence: "While at the beach, the dog eats the biscuits in the box."



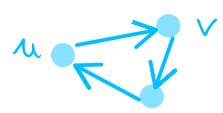
Like in undirected graphs, a loop is an edge with its two endpoints being equal. Multiple edges are edges having the same tail and the same head.

$u \rightarrow v$ Not a multiple edge

A directed graph is simple if there is no loop nor multiple edges.

Example: All the directed graphs above are simple.

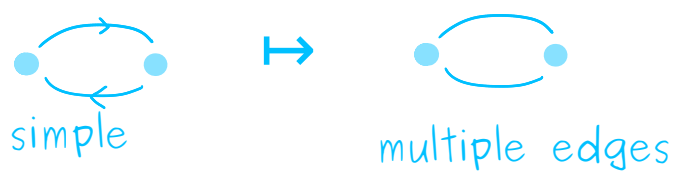
In a simple digraph, we write the edge from u to v as uv (and so this is not the same as vu). If uv is in the graph, v is a successor of u and u is a predecessor of v .



A simple digraph is a path if its vertices can be ordered so that v follows u in the vertex ordering if and only if there is an edge from u to v . The only vertex that can be repeated is the first and the last vertices, if they are equal; the path is then a cycle. Equivalently, we can define walks and trails (walks without repeated edges) in the same way as in undirected graphs.

The underlying graph of a digraph D is the undirected graph G in which we removed the orientation of the edges. Hence, $uv=vu$, and if uv and vu both appear in D , uv is a multiple edge of G .

Remark: The underlying graph of a simple digraph is not always a simple graph.



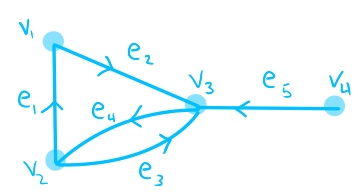
Subgraphs and isomorphisms are defined in the same way as for undirected graphs.

Adjacency and incidence matrix

In a digraph, the adjacency and incidence matrices are not defined in the same way as in graphs.

The adjacency matrix $A(D)$ of a loopless digraph D has u,v -entry the number of edges from u to v . The incidence matrix has v,e -entry $+1$ if v is tail of e , -1 if it is its head, and 0 if v is not an endpoint.

The digraph on the left has the following adjacency and incidence matrices:



$$A(D) = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \text{ The adjacency matrix is not more symmetric!}$$

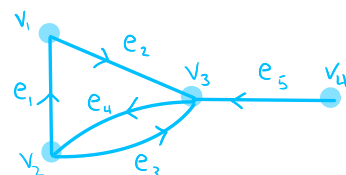
$$M(G) = \begin{pmatrix} -1 & +1 & 0 & 0 & 0 \\ +1 & 0 & +1 & -1 & 0 \\ 0 & -1 & -1 & +1 & -1 \\ 0 & 0 & 0 & 0 & +1 \end{pmatrix}$$

Connectedness: weak and strong

3

A digraph is weakly connected if its underlying graph is connected. It is strongly connected if there is a path from u to v , for every two vertices u and v .

The graph below is weakly connected, but not strongly connected, as there is no path from vertex 3 to vertex 4.



The strong components of a digraph are its maximal strongly connected subgraphs.

Degree and neighborhood, in and out

Let v be a vertex in a directed graph. Its outdegree is the number of edges that have v as a tail, and is noted $d^+(v)$. The indegree is the number of edges that have v as a head, $d^-(v)$.

The number of edges is $\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v)$.

The out-neighborhood of v is the set of vertices $\{u : vu \text{ is an edge}\}$.

The in-neighborhood is defined similarly.

Eulerian graphs

A digraph is Eulerian if it contains an Eulerian circuit, i.e. a trail that begins and ends in the same vertex and that walks through every edge exactly once.

Obviously, a graph will not be Eulerian if it has more than one nontrivial component or if the sum of the in and out degree of some vertex is odd. The following theorem gives a classification of Eulerian digraphs.

Theorem

A digraph is Eulerian if and only if there is at most one nontrivial strong component and, for every vertex v , $d^+(v) = d^-(v)$.

Proof

(\Rightarrow) If there is an Eulerian circuit, it visits all the vertices in a nontrivial component, so there is at most one of them. Also, the Eulerian circuit goes in and out of v the same number of times, which

means the in- and outdegrees must be equal.

(\Leftarrow) We prove by induction on the number of edges that if the in- and outdegrees are the same at every vertex in a strongly connected graph, there is an Eulerian circuit.

Base case: When there is no edge, the empty circuit is Eulerian.

Induction hypothesis: Suppose that, whenever there is at most m edges, every graph that has, at each vertex, the same in- and outdegree, and that has at most one non-trivial strong component, is Eulerian.

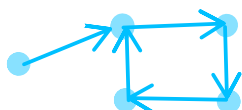
– For a graph with $m+1$ edges, we first prove the following lemma:

Lemma

If the outdegree of every vertex is at least 1, then the digraph has a cycle.

Proof (of the lemma)

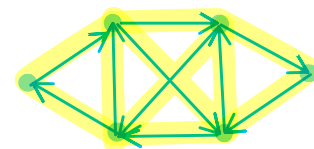
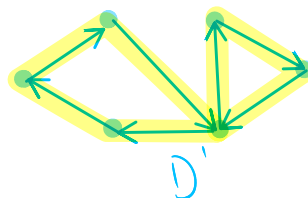
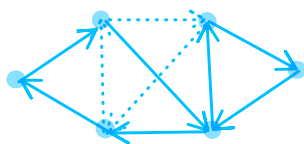
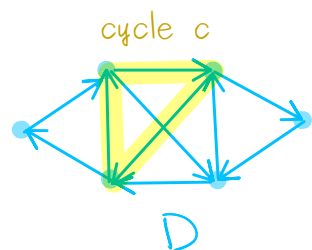
Let v be a vertex. Since it has outdegree at least 1, there is a walk starting at v . Since every vertex has outdegree at least 1, the walk can always be extended. Since the number of vertices in D is finite, the walk will go back to a vertex it already visited. The first time this happens, the part of the walk between the two occurrences of a vertex is a cycle.



Proof of the theorem (continued)

For a graph with $m+1$ edges, consider the unique nontrivial strong component. The lemma applies to it, so there is a cycle c . Removing the edges of c to the digraph preserves the equality of the in- and outdegrees. Let D' be that reduced graph. We can apply the induction hypothesis to get an Eulerian circuit in each strong component of D' . Each such component shares at least one vertex with c , since they are in the same strong component of D . To build an Eulerian circuit, we travel through c . Each time we get to a vertex that has neighbors not in c , we visit all the edges in its strong component: we know it is possible since the component is Eulerian. That process gives an Eulerian circuit in the original digraph.

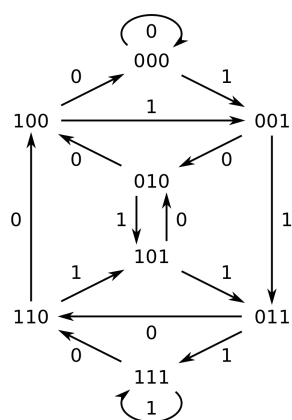




Application: De Bruijn cycles

Let D_{n+1} be the following digraph:

- vertices are binary sequences of length n
- there is an edge from a sequence s to another s' if the $n-1$ last letters of s are the $n-1$ first letters of s' .



As an example, D_4 is illustrated on the left.

Proposition

The De Bruijn graphs are Eulerian.

For the proof, use the previous theorem and verify the equality of the out- and indegrees.

Problem: What is the minimum length for a sequence containing all the binary sequences of length n ? To solve this problem, use the above proposition and your homework.

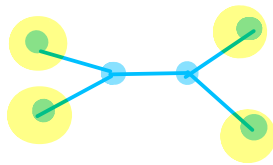
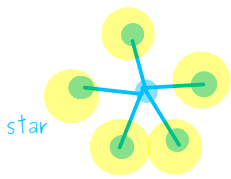
We introduce the notion of trees, a very important type of graph. Over the next week or two, we will study the properties of trees and forests.

Definition

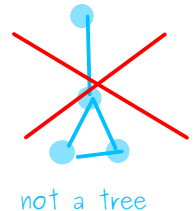
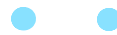
A graph with no cycle is acyclic.

An acyclic graph is a forest; a connected forest is a tree.

A leaf is a vertex of degree 1 in a tree.



a forest



not a tree



Every component here (except the one with a red X on it) is a tree, and the whole thing (without the one with the X) is a forest.

Leaves are highlighted. A star is the tree in which there is one vertex adjacent to every other.

Caveat: As graphs, trees don't need to have one specific root. We can always distinguish one root, but it is not needed. We will go back to this subject later.

Lemma

Every tree with at least two vertices has at least two leaves.

Deleting a leaf from an n -vertex tree produces a tree with $n-1$ vertices.

Proof

A tree is always connected so there is a path p between any two vertices $\{u, v\}$. Since there is no cycle, that path can only be extended finitely many times without returning to a previously visited vertex. The last time it can be extended in one direction, that vertex is a leaf, as there is no cycle.

When one deletes a leaf u from a tree, it does not disconnect it, (2) since there is no path going through that vertex (not as an endpoint), i.e. for v, w in the graph, there is no path passing through u from v to w . ■

One consequence of that lemma is that we can build every tree with at least two vertices by "adding leaves". We will discuss that topic on Wednesday.

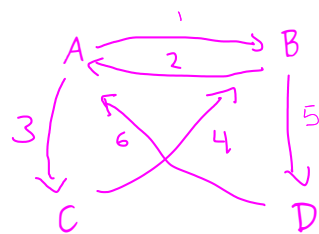
The following theorem gives multiple characterizations of trees:

Theorem

Let G be a graph with n vertices ($n \geq 1$). The following statements are equivalent:

- (A) G is connected and has no cycles.
- (B) G is connected and has $n-1$ edges.
- (C) G has no cycles and $n-1$ edges.
- (D) G has no loop and has, for each pair of vertices $\{u, v\}$, exactly one uv -path.

The proof of such a statement is a closed walk that visits every vertex in the complete digraph with vertices A, B, C and D :



Proof

(1 $A \Rightarrow B$) We need to prove that if G is connected and has no cycle, it has $n-1$ edges.

By theorem from 4/8, it must have at least $n-1$ edges for it to be connected. To prove there is at most $n-1$ edges, we prove that a graph with n edges has a cycle (which is not permitted, by hypothesis). This proof is by induction on n (the number of vertices):

Base case: If $n=1$, the edge is a loop and that is a cycle.

Induction hypothesis: Assume a graph with k vertices and k edges has a cycle.

Induction step: We need to prove that a connected graph with $k+1$ vertices and $k+1$ edges has a cycle. If there is a leaf, remove it and delete the incident edge; applying induction hypothesis tells us that there is a cycle in the rest of the graph.

If there is no leaf, then the lemma from page 1 proves the graph is not a tree.

(2 $B \Rightarrow A$) We need to show that if G is connected and has $n-1$ edges, it has no cycle. We prove the contraposition: if G is connected and has a cycle, there is more than $n-1$ edges.

Since G has a cycle, there is at least an edge that is not a cut-edge (by the theorem from 4/1). Deleting that edge would mean the graph has one fewer edge and is connected, which means, by theorem from 4/8, that the graph with one fewer edge has at least $n-1$ edges. So the original graph has at least n edges.

Until now, we proved $A \Leftrightarrow B$. They are equivalent, so we can use them together from now on.

(3 $A \Rightarrow C$) Since $A \Leftrightarrow B$, we know that G is connected, has no cycle and has $n-1$ edges, which already proves C).

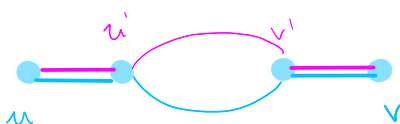
(4 $C \Rightarrow B$) We want to show that if a graph has $n-1$ edges and no cycles, it is connected. We look at each of the k components.

In the component i ($1 \leq i \leq k$), assume there are $n(i)$ vertices. Since the component is connected and has no cycle, it has $n(i)-1$ edges (by $A \Rightarrow B$). Hence, the total number of edges is

$\sum_{i=1}^k (n(i)-1) = \sum_{i=1}^k n(i) - k = n - k$. However, the hypothesis of C is that the graph has $n-1$ edges. So there is exactly one component, and the graph is connected.

Now, A , B and C are equivalent. That means that two characteristics among connectedness, no cycles and $n-1$ edges are sufficient to show a graph is a tree.

(5 $B \Rightarrow D$) Since $B \Rightarrow A$, the graph has no cycle; in particular, it has no loop. It is also connected, so there is a path p between any pair of two vertices $\{u, v\}$. To show uniqueness of that path, we use the hypothesis that there is no cycle, and contradiction: Assume there exist 2 paths p and q between u to v .



Let u' be the first vertex in p and q whose next edges differ, and let v' be the next vertices that appear both in p and q . Then, the part of p between u' and v' and the part of q between u' and v' are paths with no common vertices that have the same endpoints; gluing them together creates a cycle. Hence, there is a unique path between u and v , for any pair of vertices $\{u, v\}$.

(\Leftarrow $D \Rightarrow A$) Since there is a path between every pair of vertices, the graph is connected. The uniqueness of the path means there is no cycles, proving A . ■

Corollary

a) Every edge of a tree is a cut-edge. (by A)

b) Adding one edge to a tree forms exactly one cycle (corollary of $A \Rightarrow B$).

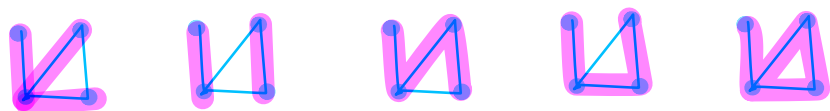
Spanning trees

Let $G=(V, E)$ be a graph.

A graph is a spanning subgraph of G if it has vertex set V .

A spanning tree is a spanning subgraph that is a tree.

Example



In purple, five spanning subgraphs of the graph in blue. Only the first, third and fourth ones are spanning trees.

Theorem

Every connected graph has a spanning tree.

Proof

Every connected graph has a connected spanning subgraph. To remove the cycles from it, delete one after the edges that are in cycles. Once there are 1 edge fewer than vertices, the graph will be a tree. ■

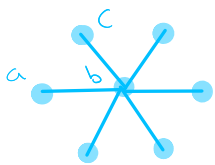
Distance in trees and graphs

5

If G has a uv -path, the distance between u and v , noted $d(u,v)$, is the smallest length of a uv -path. If G has no such path, $d(u,v)=\infty$. The diameter of G , $\text{diam}(G)$, is the maximum distance between two vertices.

The eccentricity of a vertex u is the distance to the furthest vertex. The radius is the minimal eccentricity.

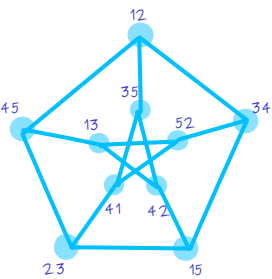
Example



Distance: $d(a,b)=d(b,c)=1$, $d(a,c)=2$

Diameter 2. A star always has radius 1, since the central vertex has eccentricity 1. The diameter, for all graph, is the maximal eccentricity.

Example



The Petersen graph has radius and diameter 2.

Recall there is an edge between two vertices if they represent disjoint 2-sets of $\{1,2,3,4,5\}$.

If two vertices ij and jk are not adjacent, they must share an element (as sets). Then, lm is disjoint from ij and jk , so $ij-lm-jk$ is a path of length 2.

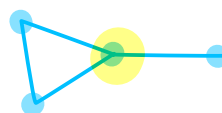
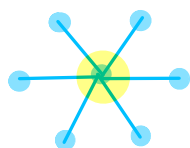
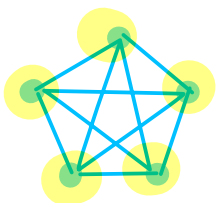
Here, $\{i,j,k,l,m\}$ represents $\{1,2,3,4,5\}$.

Theorem

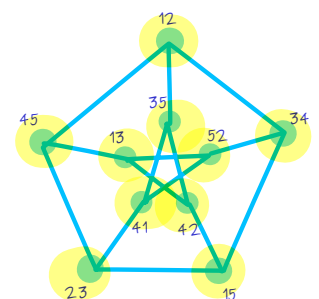
If G is a simple graph, $\text{diam}(G) \geq 3 \Rightarrow \text{diam}(\bar{G}) \leq 3$.

Proof: Read and understand as homework. In the book, that is Theorem 2.1.11, p.71.

The center of a graph is the induced subgraph with vertices of minimum eccentricity.



The Petersen graph and complete graph have center the whole graph. The star has, as center, only the central vertex.

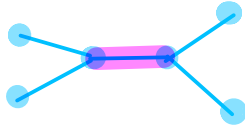


Theorem (Jordan, 1869)

6

The center of a tree is a vertex or an edge.

That means it cannot be a set of vertices, whenever the graph is a tree. Of course, the examples above show it is not true for graphs in general.



Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 2.1

Math 38 – Graph Theory

Enumeration of trees

Nadia Lafrenière
04/18/2022

How many labeled trees are there? And up to isomorphism? This is the questions we want to ask today. We will only be able to solve one of these.

Example

The number of labeled trees with $\{0,1,2,3\}$ vertices are, respectively, 1,1,1,3.

$n=0$, 1 tree

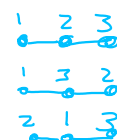
$n=1$, 1 tree



$n=2$, 1 tree

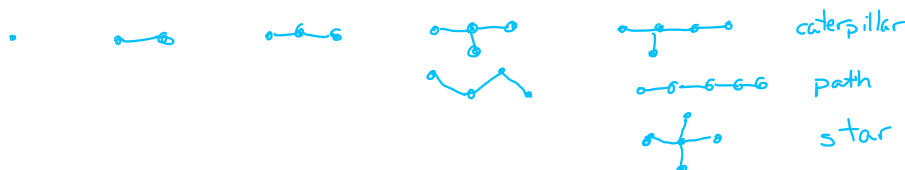


$n=3$, 3 trees



For the example with $n=3$, the three trees are isomorphic. However, as labeled trees they are not the same; their adjacency matrices are different (the vertex with degree 2 is not the same).

In the case of unlabeled trees, there are 1,1,1,1,2,3 trees of 0,1,2,3,4 and 5 vertices, respectively. These are the trees up to isomorphism.



Doing the same exercise with labeled graphs, we find there are $2^{\binom{n}{2}}$ graphs. (Proof will be in homework).

so there are at most $2^{\binom{n}{2}}$ labeled trees with n vertices. But we can expect this number to be much less.

Remark: Counting trees, or graphs, up to isomorphism is incredibly difficult! Hence, we will focus on labeled trees and labeled graphs.

Theorem (Cayley's formula; proof is from Prüfer, 1918)

There are n^{n-2} labeled trees with n vertices, if $n \geq 1$.

Proof

This proof is done using a bijection. To do so, we have to find another collection of objects indexed by the positive integers, so that there are n^{n-2} items indexed by n .

Natural choice: the n -ary sequences of length $n-2$.

Given a tree, the corresponding sequence is the Prüfer sequence or Prüfer code.

We present two algorithms: The first one takes a tree as input and transforms it into a unique n -ary sequence of length $n-2$; the second one takes an n -ary sequence of length $n-2$ and builds a unique tree with n vertices. This will prove that the number of n -ary sequences of length $n-2$ and the number of binary trees with the same vertices are the same.

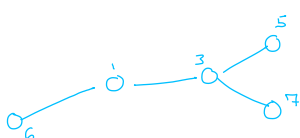
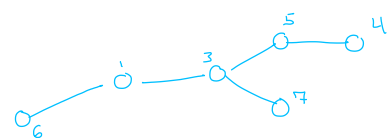
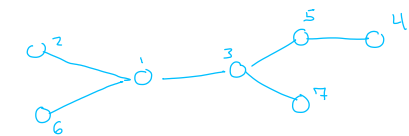
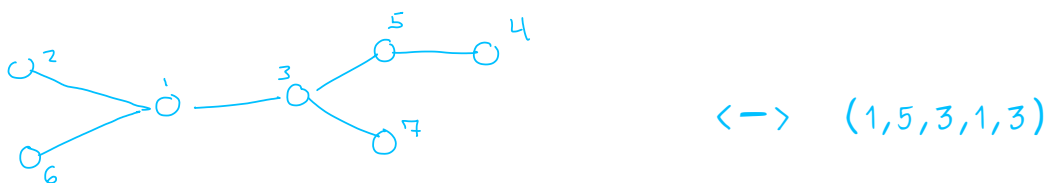
Algorithm 1: Production of the sequence

Let T be a tree with vertices $\{1, 2, \dots, n\}$. If $n < 2$, we already checked that there is a single tree. So assume $n \geq 2$.

While there is more than two vertices, remove the leaf with smallest label (it always exists, by Lemma from Friday). To the sequence, append the neighbor of that leaf (it is unique, since a leaf has degree 1).

Once there are only two vertices left, stop.

Example



stop

Algorithm 2: Production of the tree

Take a n -ary sequence s of length $n-2$, $n \geq 2$.

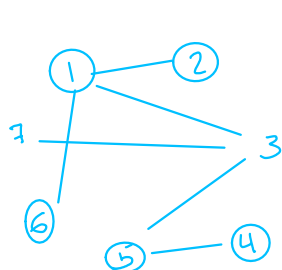
Draw n isolated vertices, and label them $\{1, 2, \dots, n\}$. We will add $n-1$ edges. At the beginning, no vertex is marked.

While the sequence is not empty:

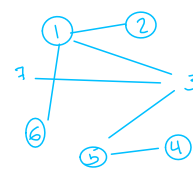
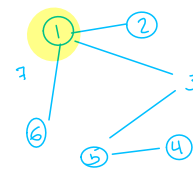
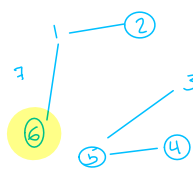
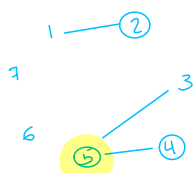
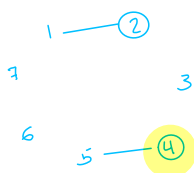
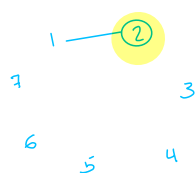
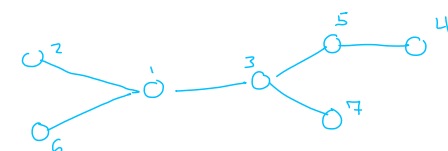
Mark the smallest unmarked vertex whose label does not appear in the sequence (this always exists, since sequence only has length $n-2$).

Delete the first element of the sequence and draw an edge between this element and the vertex you just marked. This adds one edge.

When the sequence is empty, there are $n-2$ edges, one for each element of the original sequence. There are two unmarked vertices, including one isolated. Draw an edge between them, then stop.



(1,5,3,1,3)
(5,3,1,3)
(3,1,3)
(1,3)
(3)



Claim: this is a tree. To prove this claim, we only need to prove either that it is connected or that it has no cycle, since we already know there are $n-1$ edges. We prove there is no cycle.

When we add an edge, we always do so between a marked vertex and an unmarked one; we then mark the unmarked one. If there was a cycle, that cycle would need to have at least one edge added with both endpoints that are marked, which is not possible. That proves the claim.

Since both algorithms are well-defined, there is a bijection between n -ary sequences of length $n-2$ and the trees with n vertices.

Note that the seemingly related problem of counting unlabeled trees is much harder... to the extent that no closed formula is known to count them. The only thing we know is an asymptotic estimate of the number of trees with n vertices when n is very big; even then, the proof is very hard and requires techniques that are far beyond the scope of this class.

Counting labeled trees with a given degree sequence

From now on, we want to count trees with n vertices, labeled $\{1, \dots, n\}$, and vertex i having degree d_i . How many such trees are there?

We use Prüfer sequences to solve this problem.

Observation: In the second algorithm, we always add an edge between the vertex we mark and the vertex that appears in the sequence (assuming we mark both vertices in the last step). So the degree of the vertex i is the number of occurrences of i in the sequence, plus 1.

Corollary

Given positive integers d_1, d_2, \dots, d_n summing to $2n-2$, there are exactly $\frac{(n-2)!}{\prod_i (d_i-1)!}$ trees with vertex set $\{1, 2, \dots, n\}$ such that vertex i has degree d_i .

Proof

Using the observation, we know that the number of such trees is the number of sequences with (d_i-1) occurrences of i , for every i in $\{1, 2, \dots, n\}$. The number of sequences of length $n-2$ with numbers all distinct is the number of permutations, this is $n!$ ($=1 \times 2 \times \dots \times n$). When a number is repeated k times, there are $k!$ fewer sequences: this is because we accounted the $k!$ permutations of these occurrences of the number.

Hence, the number of sequences of length $n-2$ with d_i-1 occurrences of i is $\frac{(n-2)!}{\prod_i (d_i-1)!}$.

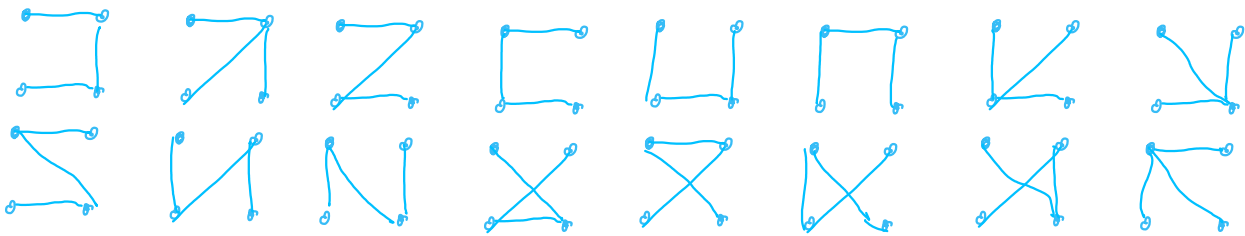


Math 38 – Graph Theory

Counting spanning trees

Nadia Lafrenière
04/20/2022

Question: How many spanning trees does the complete graph with n vertices have?



For the complete graph, there is an easy way of answering: This is the total number of trees with n vertices, as they are all subgraphs of the complete graph. Hence, it is n^{n-2} .

However, the following question is much harder:

Question: Given any graph G (simple or not), how many spanning trees are subgraphs of it?

Finding a closed formula to count the number of spanning trees would be a lot to ask for trees that don't have a specific structure. Instead, we see an algorithm to answer this question easily.

Proposition

There are as many spanning trees in a graph G as in the graph obtained from G by deleting all its loops.

Proof

Loops cannot belong to any tree, as they are cycles. So deleting them won't remove any subtree.

However, we cannot use the same strategy for multiple edges, as there can be more than one associated spanning tree. Here is an example:



Example: Count the number of spanning trees of the kite ($K - e_4$)

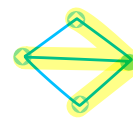
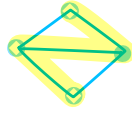
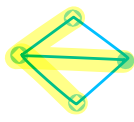
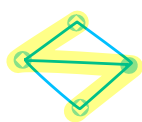


4 trees passing through the outer cycle (of length 4).



4 trees passing through the diagonal, since we need to choose one edge from each triangle.

②



So there are 8 spanning trees in the kite.

To count the number of trees, there are two cases. These two cases span all the possibilities: either we use one specific edge or we don't use it. Of course, we cannot be in both situations at the same time. Combinatorially speaking, that means the total number of spanning trees is

$\#(\text{spanning trees using that edge}) + \#(\text{spanning trees not using it})$.

If the latter seems easy to count in general, the former needs the introduction of the following operation.

In a graph G , the contraction of the edge $e=uv$ is the replacement of both vertices u and v by a single vertex, by keeping all the edges incident to it, except e . The resulting graph, $G \cdot e$, has one fewer edge than G , and one fewer vertex.

Example

In the kite, the contraction of the central edge gives the following:



Proposition

The number of spanning trees of G , noted $\tau(G)$, satisfies, for any single edge e , $\tau(G) = \tau(G - e) + \tau(G \cdot e)$.

Proof

We already noted above that the total number of spanning trees is the sum of the trees with and without edge e . The thing we need to prove is that $\tau(G \cdot e)$ is the number of spanning trees using edge e .

Start with T a spanning tree of $G \cdot e$; T is connected to the new vertex created from the contraction of e . Replacing that vertex with the edge e (and distributing the edges among the two vertices like in the G) gives a spanning tree of G using e .

Also, from any spanning tree of G using e , we get a spanning tree of $G \cdot e$ by contracting vertex G (i.e. the spanning graph is still connected and still has no cycle).



This proposition will be the key to count, recursively, the number of spanning trees. We could also benefit from some shortcuts, like the following proposition:

Proposition

If G has no loop and does not have cycles of length at least 3, its number of spanning trees is the product of the multiplicities of the edges.

Proof

Since G has no loops nor cycles of length at least 3, all the cycles have length 2, i.e. they are multiple edges. At most one of them can appear in a given spanning tree. Also, at least one of them must appear: otherwise the graph would be disconnected. This is because these edges are all not part of a cycle that uses other edges. Hence, we have to pick exactly one edge per pair of endpoint. These choices all being independent, we multiply their numbers.

Corollary

If there are k edges $\{e_1, e_2, \dots, e_k\}$ between endpoints u and v in G , the number of spanning trees of G is given by

$$\tau(G - \{e_1, e_2, \dots, e_k\}) + k\tau(G \cdot e),$$

where $G \cdot e$ is the graph obtained by merging u and v and deleting $\{e_1, e_2, \dots, e_k\}$.

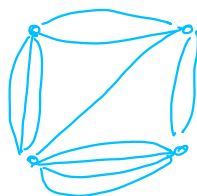
That yields an algorithm to count the number of spanning trees in G :

- If G is disconnected, it has no spanning tree; if G has a single vertex, it has only one spanning tree.
- Delete all loops in G .
- If G has no cycles of length at least 3:
 - The number of spanning trees is the product of the multiplicities of edges.
- Otherwise, choose a (multiple) edge e with multiplicity k , that is in a cycle of length at least 3. The number of spanning trees is $\tau(G - e) + k\tau(G \cdot e)$.

In the last step, $G - e$ has fewer cycles than G , and $G \cdot e$ has shorter cycles. That means that the algorithm eventually terminates.

Example

We count the spanning trees in the graph below:

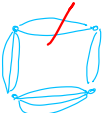


As we did earlier with the kite, we consider deleting or contracting the central edge.

Since it has multiplicity 1, the number of spanning trees can be counted in this way:

Contraction  30 spanning trees using that edge.


Deletion  # spanning trees of  $24 + 2 \times 26 = 76$

Deletion  No cycle of length ≥ 3
There are $2 \times 3 \times 4 = 24$ spanning trees not using top and diagonal edges.

Contraction $2 \times$  We need to count the number of spanning trees of the multigraph on the left, and multiply it by 2.

spanning trees of  $12 \times 2 \times 7 = 26$

Deletion  There are $3 \times 4 = 12$ spanning trees not using the right edge.

Contraction 2×7  For each edge on the right, there are 7 edges in the contracted graph.

Total number of spanning trees
 $30 + 1(24 + 2(12 + 2 \times 7)) = 106$ spanning trees

That process works for small trees, but the recursive procedure makes it very long to do for large connected graphs. We will see next class a theorem to make this computation efficient.

We give a more efficient way of counting the number of spanning trees in loopless graphs. As a second part, we are wondering if it is possible to decompose a graph into multiple copies of the same tree.

Counting spanning trees, efficiently

Last lecture, we counted the number of spanning trees using the deletion-contraction process. That felt like a good process since it allowed us to count (for the first time) the number of spanning trees. However, the algorithm to do so has an exponential complexity (i.e. the number of steps required to make it work might be as big as (roughly) $2^{|E|}$).

The following theorem gives an efficient computation for the number of spanning trees.

Theorem

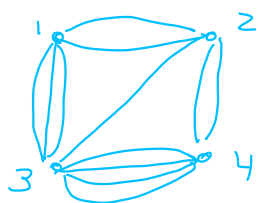
Let G be a loopless graph and A be its adjacency matrix. Let L be the matrix with $l_{ij} = -a_{ij}$ and $l_{ii} = d(i)$, the degree of vertex i .

The number of spanning trees of G is any cofactor of L .

(Recall that the (i,j) -cofactor of the matrix M is computed by $(-1)^{i+j} \det(M_{ij})$

where M_{ij} is the matrix obtained from M by deleting its i -th row and j -th column.)

Example



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

$$L = \begin{pmatrix} 5 & -2 & -3 & 0 \\ -2 & 5 & -1 & -2 \\ -3 & -1 & 8 & -4 \\ 0 & -2 & -4 & 6 \end{pmatrix}$$

$$L_{3,3} = \begin{pmatrix} 5 & -2 & 0 \\ -2 & 5 & -2 \\ 0 & -2 & 6 \end{pmatrix}$$

$$\det(L_{3,3}) = 5 \cdot (30 - 4) + 2 \cdot (-12) + 0 = 106$$

Just as we obtained last lecture, there are 106 spanning trees.

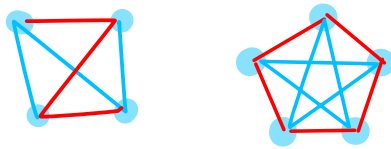
If you are interested in reading it, the proof can be found on pages 86–87 of the textbook. ²

Decomposition

Recall that a decomposition of a graph is a list of subgraphs in which every edge appears exactly once. This definition raises the following problem: When can we decompose a graph G into copies of H ?

Example

Two copies of a self-complementary graph is a decomposition of a complete graph.



Proposition

If G decomposes into many copies of H , then

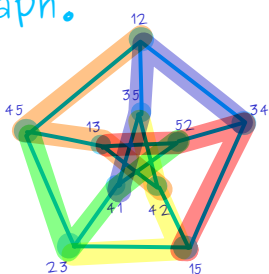
- 1) The number of edges in H divides the number of edges in G .
- 2) The maximum degree of H cannot be greater than the maximum degree of G .

Proof

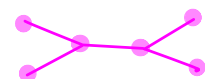
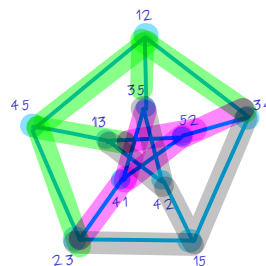
- 1) Assume there are m copies of H in G . Then, the number of edges in G is m times the number of edges in H .
- 2) Assume $\Delta(H) > \Delta(G)$. So there is a vertex v in H , and that vertex must appear in G as well. The copy in G may have more edges incident to it, but cannot have fewer. So v in G has degree $\Delta(H)$, contradicting the maximality of $\Delta(G)$.

Are these two conditions sufficient for graphs to decompose into multiple copies of a graph? The following example will show this is not enough.

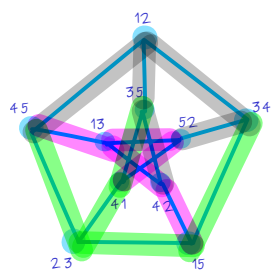
Example: Decomposition of the Petersen graph. The triangle (3-cycle) satisfies conditions 1) and 2), but does not appear in the Petersen graph.



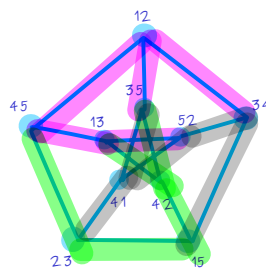
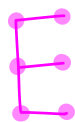
path



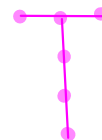
"H-graph"



"E-graph"



"T-graph"



The Petersen graph decomposes into multiples copies of the following graphs:

- 1 copy of itself;
- 15 copies of an edge;
- 3 copies of either the H-graph, the E-graph or the T-graph;
- 5 copies of paths of length 3.

By the proposition above, 1, 3, 5 and 15 are the only possible numbers of edges that can appear in the smaller graphs. To prove the list is exhaustive, we must show that the H-graph, the E-graph and the T-graph are the only graphs with 5 edges that can occur in the decomposition, and the same has to be true for the path of length 3 compared to other graphs of size 3.

For the case of 5 edges:

- The Petersen graph is regular of degree 3, so no vertex can have degree 4 or 5 (remark 2).
- The shortest cycle in the graph has length 5. The graph cannot be decomposed into cycles of length 5, because vertices would need to have even degree.
- The graph cannot be decomposed into paths of length 5, since there would be at most 6 vertices of odd degree (the endpoints of the path.)
- So the smaller graph has no cycle, at least 3 leaves and maximal degree at most 3. The possibilities are the following:



To decompose it into 5 copies of a graph, with 3 edges, there are 3 options:



- The triangle is not possible since it does not appear in the Petersen graph;
- The claw (star) is not possible because of the following proposition:

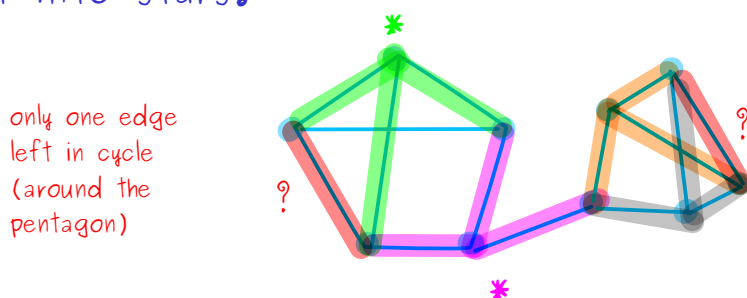
Proposition

A k -regular graph can be decomposed into copies of stars with k edges if and only if the graph is bipartite.

Proof

\Leftarrow If the graph is bipartite, consider one of the two independent sets in the bipartition. Every vertex in this set has degree k , so that vertex and the incident edges form a star with k edges. Also, every edge of the graph appears in exactly one of these stars, since every edge has exactly one endpoint in this independent set.

\Rightarrow We prove the contrapositive: If a graph is not bipartite, it cannot be decomposed into stars with k edges. Assume it is not bipartite, so it contains at least one odd cycle. In this cycle, every other vertex must appear as the center of the star; otherwise, there are edges that cannot appear in the decomposition (like the ones in red below). Also, since every edge appears once, there cannot be two neighbouring vertices that appear, because every vertex takes all k the incident edges. If two adjacent vertices appeared, there would be an edge counted twice. So there cannot be an odd cycle in the graph to decompose it into stars.



This proposition allows us to conclude with the only possible decompositions for the Petersen graph.

Graceful labelings

In general, the problem of decomposing a graph into many copies of graphs is a very hard one. Even the easier problem of decomposing it into trees is hard, as shown by the following conjectures:

Conjecture (Ringel, 1964)

If T is a fixed tree with m edges, then K_{2m+1} decomposes into $2m+1$ copies of T .

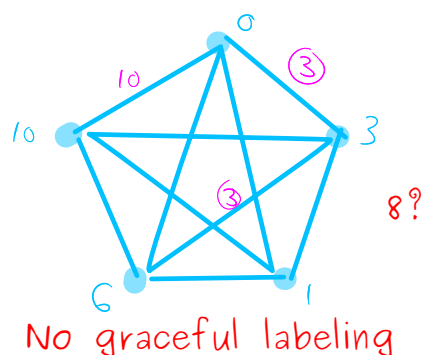
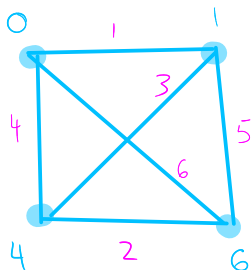
Note that despite multiple attempts to prove this conjecture, it is still open. Most attempts to solving it focus on graceful trees.

Definition

A graceful labeling of a graph G with m edges is a labeling of a graph with the numbers $\{0, \dots, m\}$ such that distinct vertices receive distinct labels and edges receive the difference of labels; the labeling is graceful if all the numbers $\{1, \dots, m\}$ appear on the edges of the graph. A graph is graceful if it has a graceful labeling.

Remark

To make it possible to define such a labeling, a graph must have at least as many edges as the number of vertices minus one (which is the case of connected graphs, for example).



Example

All stars and paths are graceful. Exercise: Find a proof!

Conjecture (Graceful Tree Conjecture – Kotzig, Ringel, 1964)
Every tree has a graceful labeling.

Theorem (Rosa, 1967)

If a tree T with m edges has a graceful labeling, then K_{2m+1} has a decomposition into $2m+1$ copies of T .

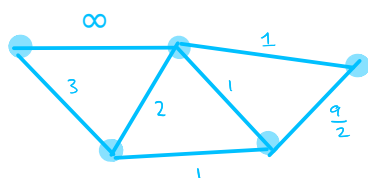
Not an
iff statement



We consider two problems, and use weighted graphs to solve them: The first one is the problem of the minimal spanning tree (where minimal refers to the weight on the edges), and the second one is the shortest path.

Weighted graphs

A weighted graph is a graph with edges labeled by numbers (called weights). In general, we only consider nonnegative edge weights. Sometimes, ∞ can also be allowed as a weight, which in optimization problems generally means we must (or may not) use that edge.



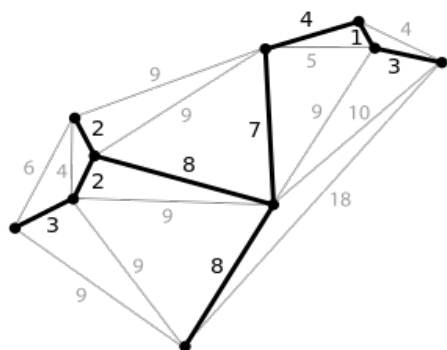
Minimum spanning tree

Problem: Given any weighted graph, find the spanning tree with the minimum weight, where the weight of a tree is the sum of the weights of its edges.

Example

An internet provider wants to wire cable in a new housing development and wants to reach every house. However, due to certain weather conditions and due to the distance between houses, the cost of reaching houses might not be the same from every path. The graph below illustrates the potential cost of every section: that is the weight of the edges, and the houses correspond to vertices.

How can they reach every house at minimum cost?



To make sure they connect every house, they must build a spanning tree. To find the spanning tree with minimal cost, they can use, for example, Kruskal's algorithm. They have no incentive to create a cycle.

Kruskal's algorithm: Given a connected weighted graph $G=(V,E)$, find its minimal spanning tree. ②

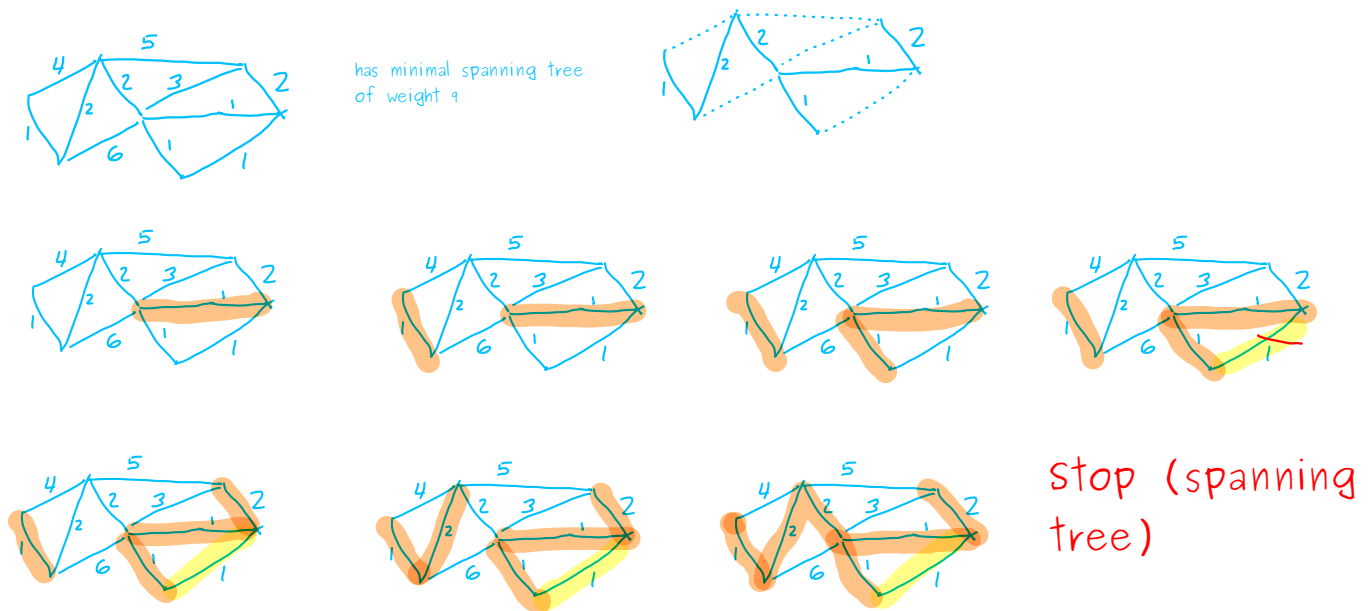
Idea: At every step, we have a forest H . We add edges from G to H until H is a spanning tree. The subgraph always stay acyclic. To ensure H is minimal, we consider edges to be added in increasing order of their weight.

Initialization: H has $|V|$ isolated vertices (no edge). The edges of G are sorted in increasing order of their weight.

Iteration: Consider the next smallest edge of G . If adding it to H reduces the number of components of H , we do so. (Otherwise, it creates a cycle, so we do not add it).

Stop: When we get at the end of the list of edges, or when H is a spanning tree, whichever comes first.

Example



Theorem (Kruskal, 1956)

In a connected weighted graph, Kruskal's algorithm constructs a minimum-weight spanning tree.

Proof

We prove the following two things:

- 1) The result is always a spanning tree.
- 2) There can't be any spanning tree with smaller weight.

1) We must show that the result is acyclic, connected and reaching every vertex. 3

- It is acyclic, since we only add edges that reduce the number of connected components. These edges cannot create cycles.
- Obviously, if we stop because the graph is a spanning tree, it is connected and reaches every vertex. Otherwise, we stop because we considered adding every edge; we did not add them only if they did not reduce the number of components. So the number of components in the forest is the same as in the original (connected) graph. Since we started with all the vertices of G , a tree is always spanning.

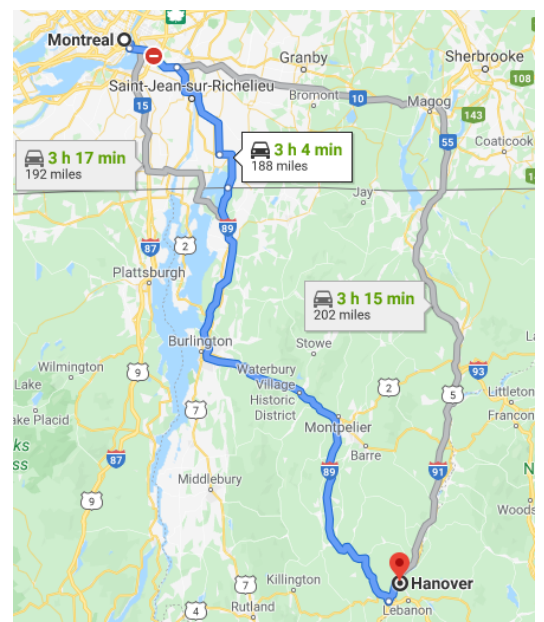
2) We prove it by contradiction. Assume T is a spanning tree with lower weight than H (obtained by the algorithm). They both have the same number of edges (since they are spanning trees), so there is at least one edge e in T but not in H . Conversely, there is an edge e' of H that is not in T . Since T has lower weight than H , we can choose $e < e'$. In fact, take the smallest such e . We considered e before e' and did not add it to H . Necessarily, e would have created a cycle in H , so there is a cycle in T (because e is the smallest edge in T but not in H). A contradiction. So $T=H$, and H is the minimal spanning tree.

Kruskal's algorithm is not the only algorithm that does so. See, for example, Prim's algorithm, where you grow a tree from one single vertex.

Shortest paths

Given two vertices in a labeled graph, what is the shortest path?

Dijkstra's algorithm gives all the distances from a given vertex u to other vertices in the graph.

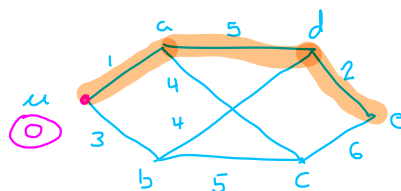
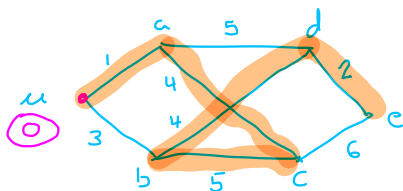


Observation: This cannot be done using a greedy algorithm.

Example

Shortest path from u to e

↑
greedy = informed by
local data



With a greedy algorithm: $d(u,e)=16$?

A (much) shorter path: $d(u,e) \leq 8$

Dijkstra's algorithm

We need a weighted graph, and we compute the minimum-weight path from one specific vertex u to every other vertices. An edge that does not exist is equivalent to an edge with weight ∞ .

Idea: The distance $d(u,v)$ is the weight of the edge between u and v if they are adjacent. We give tentative distance $d(u,w)$ for every vertex w not adjacent to u , and that distance never increases during the process.

Initialization: The set of visited vertices is $\{u\}$, $d(u,u)=0$, the tentative distance from u to x , $t(x)$, is the weight of the edge between u and x (∞ if it does not exist).

Iteration: Take the vertex x with shortest value of $t(x)$ amongst the non-visited vertices. The distance to u is $d(u,x)=t(x)$.

"Visit it" by modifying $t(y)$ for all its neighbors, by the minimum of these:

- what $t(y)$ was already; the tentative distance does not change
- $d(u,x) +$ the weight of the edge xy ; there is a shorter path

stop when you visited every vertex (for a connected graph).

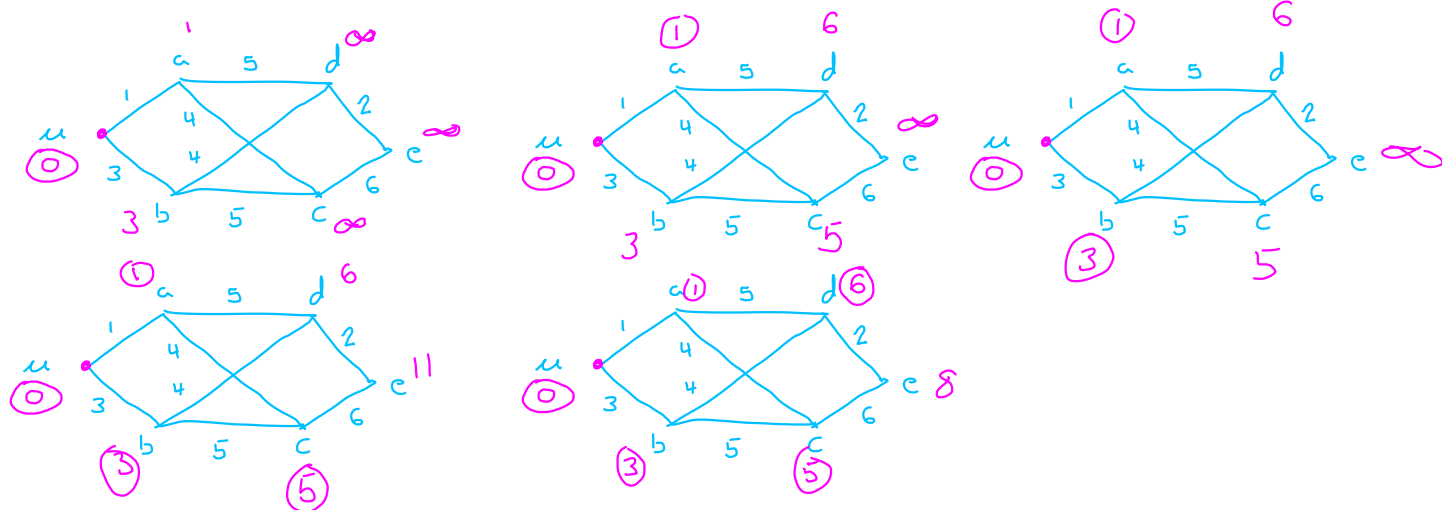
Theorem

Dijkstra's algorithm computes the distance $d(u,x)$ for every vertex x in a connected graph.

Sketch of proof

- Previously visited vertices cannot see their distance increase.
- When a vertex is visited, there cannot be a shorter path passing through a non-visited vertex.

Example



Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 2.3

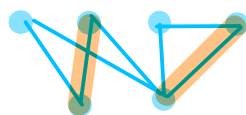
Math 38 – Graph Theory

Maximum and perfect matchings

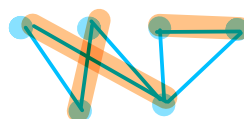
Nadia Lafrenière
04/27/2022

In a graph, a matching is a subgraph with maximal degree 1 (so every vertex is connected to at most one other vertex).

A maximal, but not maximum matching



A maximum matching, also a perfect matching

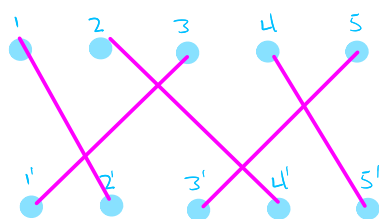


A vertex that appears in a matching is saturated, otherwise it is unsaturated.

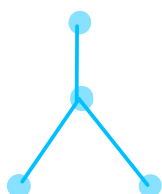
A perfect matching in a graph is a matching that saturates every vertex.

Example

In the complete bipartite graph $K_{m,n}$, there exists perfect matchings only if $m=n$. In this case, the matchings of graph K_n represent bijections between two sets of size n . These are the permutations of n , so there are $n!$ matchings.



- Perfect matchings can only occur when the number of vertices is even.
- That is not a sufficient condition, as shown by the claw.



No possible perfect matching, since the center vertex is saturated by any edge.

Example

Counting the perfect matchings in a complete graph.

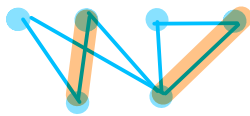
- K_n has no perfect matching if n is odd.
- Otherwise, it has $(n-1)(n-3)\dots 3\cdot 1$ perfect matchings:
 - Label the vertices $1, \dots, n$
 - Match vertex 1 with any of its neighbors; there are $n-1$ possible choices
 - As long as there are still unsaturated vertices, match the smallest unsaturated vertex with another one. The number of ways to do so is $n-3$, then $n-5$, ..., until there is only one way to do so.

Maximum matchings

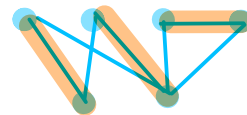
A matching of a graph is maximal if no edge can be added. It is maximum if no other matching of this graph has more edges than it.

Example

Maximal



Maximum

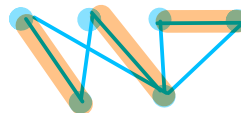
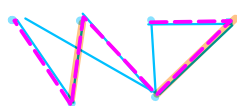


Perfect \Rightarrow Maximum \Rightarrow Maximal

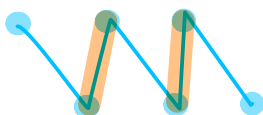
In general, the converse is not true.

Can we transform a maximal matching into a maximum matching?

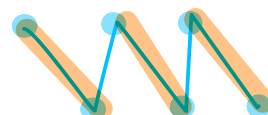
M-alternating path



Let G be a graph and M be a matching of G . An M-alternating path is a path of G that alternates between edges in M and edges not in M . An M-augmenting path is an M-alternating path with both endpoints unsaturated.



augmentation



Remark: When M is maximum, there is no augmenting path.

Theorem (Berge, 1957)

A matching M in a graph is a maximum matching if and only if the graph has no M -augmenting path.

Proof

\Rightarrow follows from the remark above

\Leftarrow We prove the converse: if it is not maximum, it has an augmenting path. If M is not maximum, then there is a matching M' with more edges.

We consider the subgraph H with edges that appear in exactly one of M and M' (not in both).

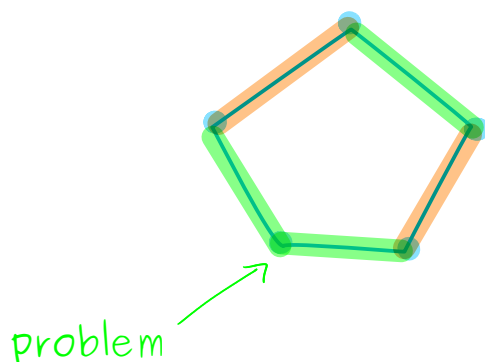
Claim: components of H are all either even cycles or paths.

- Even cycles have as many edges from M as from M' . This is because every endpoint can have at most one edge in M and one in M' .
- Since $|M'| > |M|$, there is at least one path in H with more edges of M' than edges of M . This is a path that starts and ends with unsaturated vertices of M , so this is an M -augmenting path.

Proof of the claim: That means that every vertex of H has degree at most 2, and that cycles have even length.

The maximal degree of H is 2 by construction. At most, one vertex can have one incident edge in M and one in M' .

If a cycle has odd length, then most edges belong to the same matching, and there must be two edges belonging to the same matching and incident to the same vertex. That contradicts the construction of a matching.

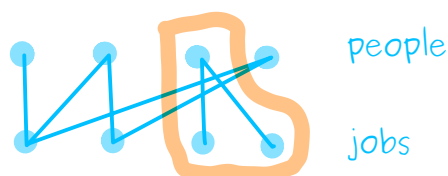


Matchings in bipartite graphs

4

Example: Job assignments

If there are m jobs and n people, not all qualified for all the jobs, can we always fill all the jobs?



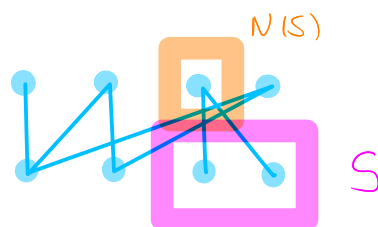
The edges are between a job and a qualified person for that job.

(The jobs cannot all be filled in this example).

Theorem (Hall's Theorem, 1935)

Let G be a bipartite graph with the independent sets X and Y forming a partition of the vertices.

G has a matching that saturates every vertex of X if and only if the neighborhood of every $S \subseteq X$ has order at least $|S|$.



Consequence: Stable marriages. Watch the video:

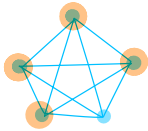
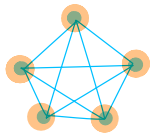
<https://www.numberphile.com/videos/stable-marriage-problem>

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 3.1

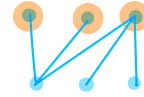
Recall from last class that a matching is a subset of edges such that no vertex appears twice as endpoints. We compare these notions with those of edge covers and vertex covers.

A vertex cover is a set S of vertices of G that contains at least one endpoint of every edge of G . The vertices in S cover G .

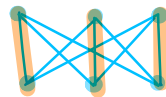
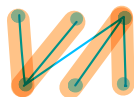
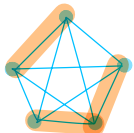
Examples



A cover of K_n has at least size $n-1$.



An edge cover is a set of edges of G that contains as endpoints every vertex of G .



The minimum number of edges in an edge cover is $\#V/2$.

Sets	Items	Property
Matchings	Edges	At most one edge per vertex
Edge covers	Edges	At least one edge per vertex
Vertex covers	Vertices	At least one vertex per edge
Independent set	Vertices	At most one vertex per edge

When are:

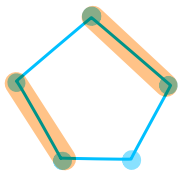
- Vertex covers and independent sets equal?
- Matchings and edge covers equal?

Matchings and vertex covers

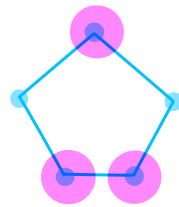
From the table above, it might seem that matchings and vertex covers are not related. However, consider a matching. In a vertex cover, every edge of the matching has to be covered by one of its endpoint. So the edge uv has either u or v (or both) in the vertex cover. Also, u (and v) cannot belong to more than one edge in the matching. So we have the following proposition:

Proposition

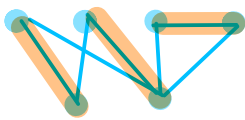
If M is a matching of G and S is a vertex cover of the same graph G , $|M| \leq |S|$.



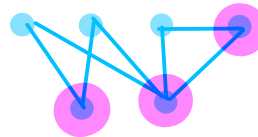
Largest matching:
size 2.



Smallest vertex cover:
size 3.



Largest matching:
size 3.



Smallest vertex cover:
size 3.

In the last example, the matching and the vertex cover have the same size ($|M|=|S|$). Of course, one can get smaller matchings and larger vertex covers, but the first example shows it is not possible to get a matching and a vertex cover of the same size.

Remark

The statements

"For any matching M and any vertex cover S , $|M| \leq |S|$ "

and

"The maximum size of a matching is always at most the minimum size of a cover"

are equivalent.

Theorem (König, Egerváry, 1931, independently)

If G is a bipartite graph, then the maximum size of a matching in G is equal to the minimum size of a vertex cover in G .

Proof

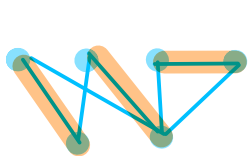
Available as, either:

- the proof of Theorem 3.1.16 in the textbook.
- the short proof of Romeo Rizzi (the paper is on Canvas).

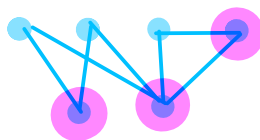
The first one is a proof by construction, the second one is a proof by contradiction.

Remark

This is not an if and only if statement. For example, the graph below contains a triangle, but has a matching a vertex cover of the same size.

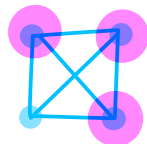
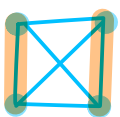


Largest
matching:
size 3.



Smallest vertex cover:
size 3.

Also, being a perfect matching is not enough, as shown by this example with just 4 vertices.



Complete graphs have vertex covers
of size $n-1$ and maximum matching
of size $\lfloor \frac{n}{2} \rfloor$

Remark

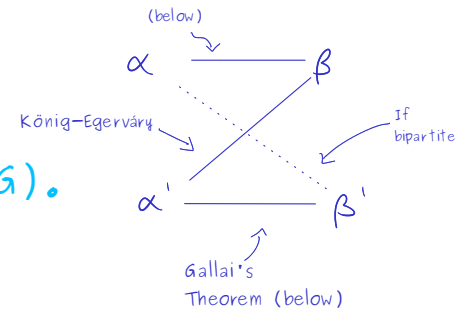
The theorem above is an example of a min-max relation: that means that solving an optimization problem calling for the minimum of something in a graph is equivalent to solving an optimization problem for the maximum. Here: the maximum matching and the minimum vertex cover. We will see more of them over the term.

Optimization problems

Sets	Items	Problem	Notation
Matchings	Edges	Finding maximum	$\alpha'(G)$
Edge covers	Edges	Finding minimum	$\beta'(G)$
Vertex covers	Vertices	Finding minimum	$\beta(G)$
Independent set	Vertices	Finding maximum*	$\alpha(G)$

* This is defined as the independence number

As an example on how to use these notations, König-Egerváry theorem states that $\alpha'(G) = \beta(G)$ for bipartite graphs. For any graph G , $\alpha'(G) \leq \beta(G)$.



Lemma ($\alpha - \beta$)

In a graph $G=(V,E)$, S is an independent set if and only if $V-S$ is a vertex cover. Hence, $|V| = \alpha + \beta$.

Proof

Let S be an independent set, meaning there is no edge between two vertices of S ; every edge has at least one endpoint in $V-S$. That means that $V-S$ is a vertex cover.

Conversely, if S' covers G , every edge has at least one endpoint in S' , so $V-S'$ is independent.



Theorem (Gallai, 1959, $\alpha' - \beta'$)

Let G be a graph with n vertices, none of them being isolated.

Then, $\alpha'(G) + \beta'(G) = n$. → size of maximum matching + minimum edge cover

Sketch of proof

2 steps:

1. From a maximum matching (of size $\alpha'(G)$), construct an edge cover of size $n - \alpha'(G)$. That implies that $n - \alpha'(G) \geq \beta'(G)$.
2. From a minimum edge cover (of size $\beta'(G)$), construct a matching of size $n - \beta'(G)$. That implies that $\alpha'(G) \geq n - \beta'(G)$.

These two steps prove that $\alpha'(G) + \beta'(G) = n$.

Corollary (König, 1916, α - β')

If G is a bipartite graph with no isolated vertex, then the size of a maximum independent set is the size of a minimum edge cover.

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 3.1

Math 38 – Graph Theory

Connectivity

Nadia Lafrenière
05/02/2022

Cuts and connectivity

A vertex cut (or separating set) is a subset of vertices S such that $G-S$ has more than one component.

The connectivity of G , $\kappa(G)$, is the minimum size of a separating set, if it exists, or $n-1$.

A graph is k -connected if its connectivity is at least k .

Examples

Disconnected = connectivity 0

Connected = 1-connected

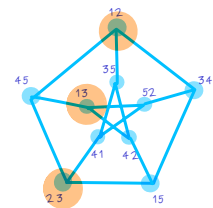
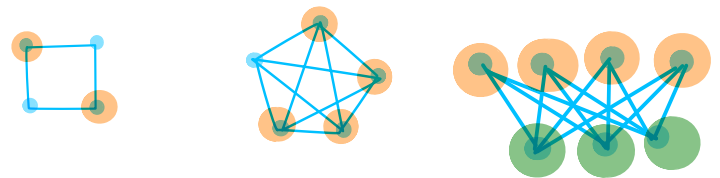
Cycles of length at least 3 have connectivity 2

Petersen graph has connectivity 3.

Complete graph K_n has connectivity $n-1$.

Complete bipartite graph $K_{m,n}$ has connectivity $\min\{n, m\}$.

By convention, we say the graph with one vertex has connectivity 0.



Proposition

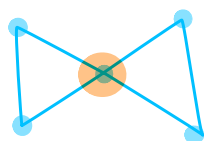
The connectivity of a connected graph is at most its minimum degree.

Proof

One can isolate a single vertex by removing all the vertices around it.

Remark

The connectivity of a connected graph is not at least its minimum degree.

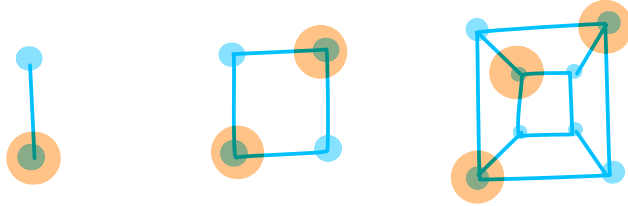


Minimum degree 2, but there is a cut-vertex \Rightarrow connectivity 1.

Example

The hypercube H_k has connectivity k .

Of course, since it is k -regular, it has connectivity at most k .
We can prove by induction it has connectivity at least k :



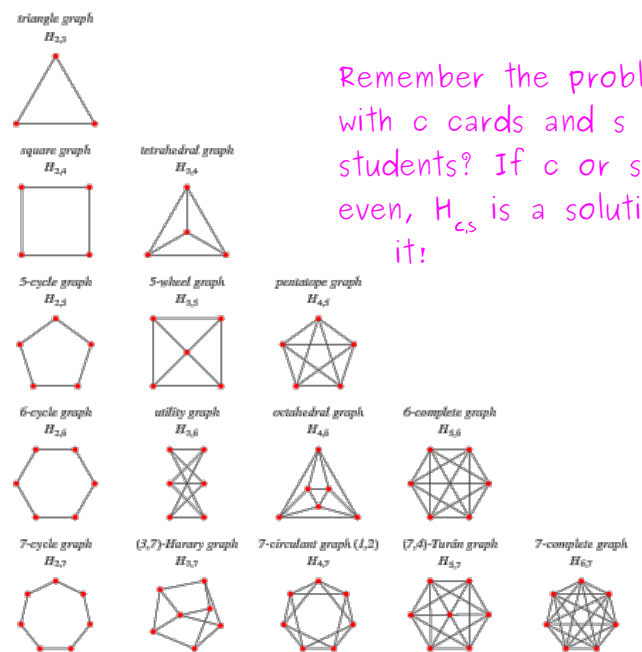
Example: Harary graphs

Harary graphs $H_{n,k}$ are graphs with n vertices and $\lceil \frac{nk}{2} \rceil$ edges, $2 \leq k < n$, being as regular as possible.

They have connectivity k :

- k is the minimum degree of $H_{n,k}$

There is a proof in the textbook that it has connectivity at least k .



Remember the problem with c cards and s students? If c or s is even, $H_{c,s}$ is a solution to it!

Theorem (Harary, 1962)

Let $k \geq 2$. The minimum number of edges in a k -connected graph with n vertices is $\lceil \frac{nk}{2} \rceil$.

Proof

This is an example of an extremal problem:

- There cannot be fewer edges in a k -connected graph. Since G is k -connected, the minimum degree is at most k . Then, there must be at least $\lceil \frac{nk}{2} \rceil$ edges.
- Example of k -connected graphs with n vertices and $\lceil \frac{nk}{2} \rceil$ edges are the Harary graphs.

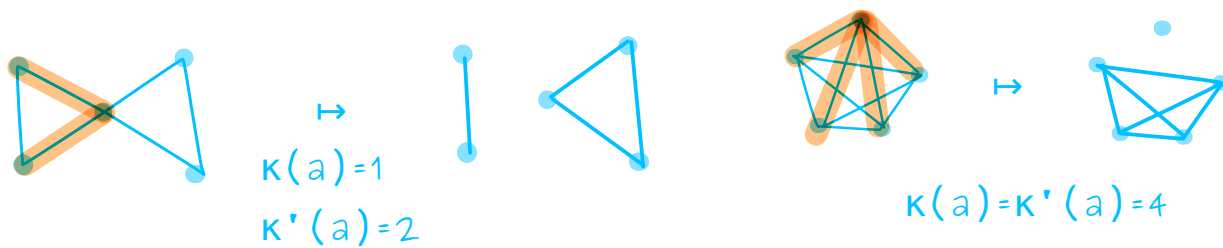
What if we instead consider the number of edges we need to remove to disconnect a graph?

Definition

A disconnecting set is a subset of edges $F \subseteq E$ such that $G - F$ has at least 2 components. separating \neq disconnecting

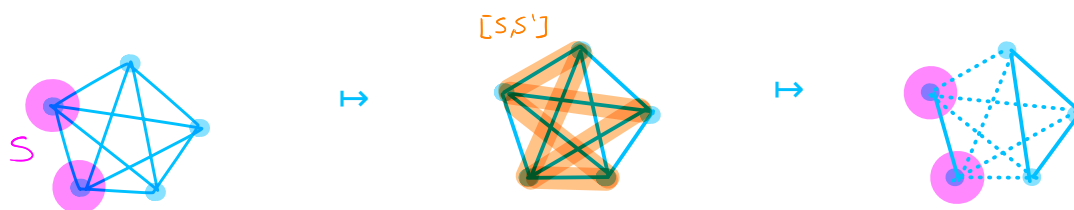
The edge-connectivity is the minimum size of a disconnecting set, and is noted $\kappa'(G)$. A graph is k -edge-connected if it has edge-connectivity at least k .

Examples

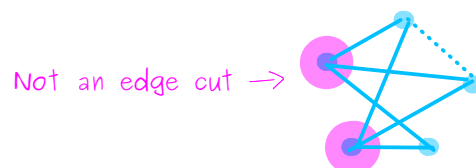


Complete graphs have edge-connectivity $n-1$. You can prove it!

Let $S \subseteq V$ be a vertex subset of a connected graph G . Let $[S, \bar{S}]$ be the set of all edges with one endpoint in S and one in \bar{S} . Then $[S, \bar{S}]$ is an edge cut.



Edge cut $\not\Rightarrow$ Disconnecting set



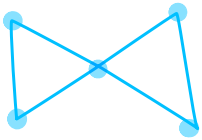
Edge cut \Leftrightarrow Minimal disconnecting set

Theorem (Whitney, 1932)

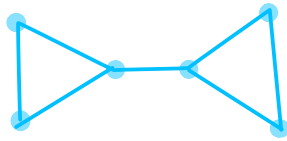
If G is simple, then $\kappa(G) \leq \kappa'(G) \leq \delta(G)$. In words: vertex-connectivity is at most edge-connectivity, which is always at most the smallest degree.

Example of inequalities

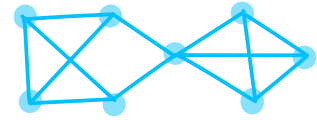
$$\kappa(G) < \kappa'(G) = \delta(G)$$



$$\kappa(G) = \kappa'(G) < \delta(G)$$



$$\kappa(G) < \kappa'(G) < \delta(G)$$



Proof

We first prove $\kappa'(G) \leq \delta(G)$. Let v be a vertex with degree $\delta(G)$. The edge cut for the set $\{v\}$ has $\delta(G)$ edges, so an edge cut with $\delta(G)$ edges exist, and the minimum edge cut has size at most $\delta(G)$.

We also need to prove $\kappa(G) \leq \kappa'(G)$. To do so, we start with a minimum edge cut, and construct a vertex cut with at most the same size. If this process is always possible, that proves the desired inequality.

Consider a minimum edge cut $[S, V-S]$. There are two cases:

- If every vertex of S is connected to every vertex of $V-S$, then $\#[S, V-S] = |S||V-S| \geq |V|-1$. Also, by definition, $\kappa(G) \leq |V|-1$.

So $\kappa(G) \leq |V|-1 \leq \#[S, V-S] = \kappa'(G)$ (the last equality is because the minimum edge-cut is the minimum disconnecting set).

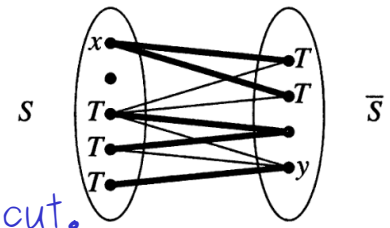
- Otherwise, there is one vertex x in S and y not in S that are not adjacent. We construct a set of vertices T :

- All neighbors of x in $V-S$.
- All vertices of $S \setminus \{x\}$ that are adjacent to vertices in $V-S$.

Then, T is a vertex cut: There is no way to go from x to y without passing through one edge of T , so $G - T$ is disconnected. We need to show that T has at most $\#[S, V - S]$ vertices.

For each vertex t of T :

- If t is a neighbor of x , then xt is in the edge cut.
- If t is in S , then t is adjacent to at least one vertex u in $V - S$. Then tu is in the edge cut.



No edge is counted twice in this list, because x is not in T .

Since every edge in this list is in the edge cut, then $|T| \leq \#[S, V - S]$, and $\kappa(G) \leq \kappa'(G)$.

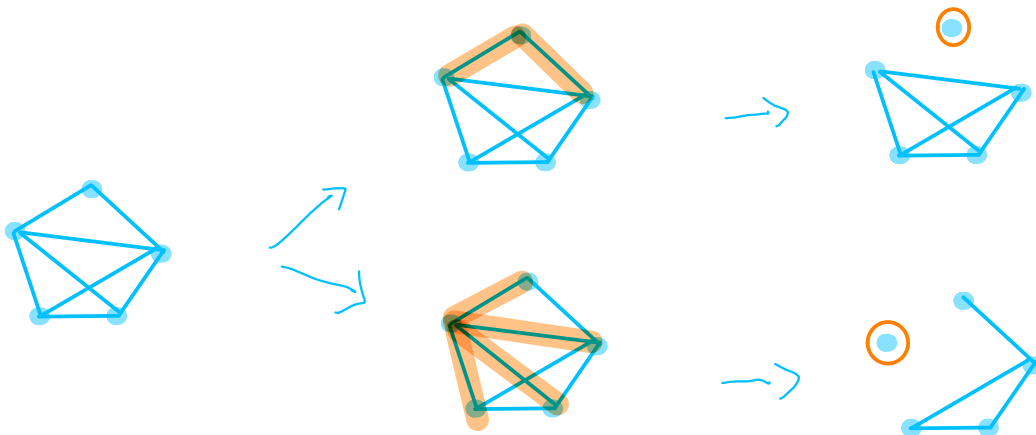


Proposition

Let G be a connected graph. Then, an edge cut F is minimal if and only if $G - F$ has exactly two components.

Remark

If we replace minimal by minimum, then the statement becomes false: $G - F$ can have two components while there are edge cuts with size smaller than $|F|$.



With your study group, try to agree on an explanation of why this is true.

Edge connectivity for regular graphs

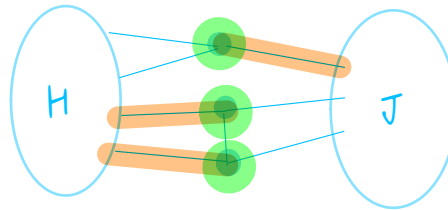
Theorem

If G is a 3-regular graph, then $\kappa(G) = \kappa'(G)$.

Proof

We already know that $\kappa(G) \leq \kappa'(G)$, in general. To prove the statement, we only need to show the reverse inequality (\geq), that is, from a minimum vertex cut, create an edge cut of the same size.

Let S be a minimum vertex cut, and let H and J be two components of $G - S$. Since S is minimum, every vertex of it has a neighbor in H and a neighbor in J . Also a vertex cannot have at least two neighbors in both H and J since G is 3-regular. For each vertex v in S , delete the edge from v to the component in which it has only one neighbor (if there is one neighbor in H , one in J and another one (in S for example), delete the edge to H).



That process breaks all the paths between H and J , so the deleted edges form an edge cut. Also, the size of that edge cut is $|S|$, which proves the statement.



We keep looking at the interconnections between edge-connectivity and vertex-connectivity. We also consider what it means for cycles and paths.

Blocks

Is a connected graph with no cut-vertex 2-connected?

Connectivity 0

Connectivity 1

Definition

A block of a graph G is a maximal connected subgraph that has no cut-vertex.



Properties

- Isolated vertices, as well as "isolated edges" (isolated copies of K_2) are blocks.
- A cycle is always 2-connected, so it is always inside the same block.
- Since the only edges that are not in cycles are cut-edges, an edge with its two endpoints is a block if and only if it is a cut-edge.
- Blocks in a tree are edges (along with their two endpoints).
- Blocks in a loopless graph are its isolated vertices, its cut-edges and its 2-connected components.

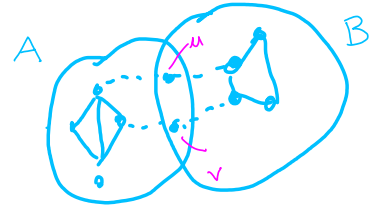
Proposition

Two blocks in a graph share at most one vertex.

Proof

By contradiction. If two blocks A and B share vertices u and v , they are connected components with no cut-vertices inside. They are also maximal, so if we extend their size, we will be creating a cut-vertex.

Since there is a path from u to v in A and one in B (because blocks are connected), there is a cycle containing u and v , and A and B form together a 2-connected component. Hence, they are in the same block.



Proposition

If two blocks share a vertex, it is a cut-vertex.

2-connected graphs

Two paths from u to v are internally disjoint if they have no common internal vertex.



Theorem (Whitney, 1932)

A graph with at least three vertices is 2-connected if and only if there exist internally disjoint u,v -paths for each pair $\{u,v\}$.

Proof

⇐ Since there are at least 2 disjoint u,v -paths for every pair $\{u,v\}$, u and v cannot be separated by removing one vertex. This is true for all $\{u,v\}$, so the graph does not have connectivity 1. It must have connectivity at least 2, and is hence 2-connected.

⇒ By induction on $d(u,v)$, the distance between u and v .

Base case: u and v are adjacent. Since the graph is 2-connected, it is also 2-edge-connected, and removing edge $e = \{u,v\}$ lets the graph connected, which means there is a path between u and v avoiding e .

Induction hypothesis: If distance is $k = d(u,v)$, there exists two internally disjoint uv -paths.

Induction step: Let u and v be at distance $k+1$, and let P be a uv -path of (minimal) length $k+1$. Let w be the vertex on P at distance k of u , so w is adjacent to v , and P' be that portion of P .

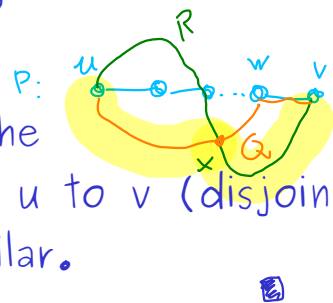


By induction hypothesis, there exist two internally disjoint uw -paths, P' and Q' .

If Q' contains vertex v , let Q be the portion from u to v in Q' ; then Q is a uv -path that is internally disjoint from P .



Otherwise, consider $G-w$. It is connected since there is no cut-vertex. So there is a path R between u and v avoiding w . If it avoids P or Q , R is internally disjoint from it. Otherwise, let x be the last vertex of R that also belongs to either P or Q . If x belongs to Q , then P is disjoint from the part of Q between u and x and from the part of R between x and v , which is a path from u to v (disjoint from P). If x belongs to P , the argument is similar.



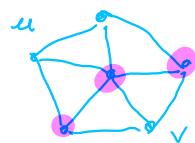
Corollary

For a graph with at least three vertices, the following conditions are characterization of 2-connected graphs:

- (A) G is connected and has no cut-vertex.
- (B) For every pair of vertices $\{u, v\}$, there are internally disjoint u, v -paths.
- (C) For every pair of vertices $\{u, v\}$, there is a cycle through u and v .

Menger's theorem

Given two vertices u and v , a uv -cut is a set of vertices S such that $G-S$ has no uv -path.



Let $\kappa(u, v)$ be the size of a minimum uv -cut.

Proposition

For u and v vertices of G , $\kappa(u, v) \geq \kappa(G)$.

Proof

A uv -cut makes the graph disconnected, so the connectivity is at most the size of a uv -cut.

Let $\lambda(u,v)$ be the maximum number of internally disjoint uv -paths.

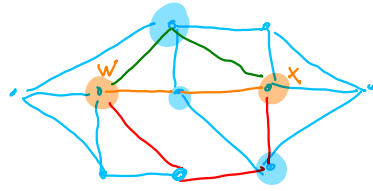
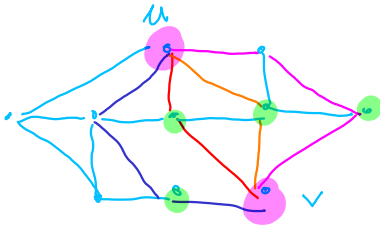
④

Proposition

For u and v vertices of G , $\kappa(u,v) \geq \lambda(u,v)$.

Proof

We need to delete at least one vertex per path, and no vertex belongs to two paths.



- Minimal uv -cut, size 4
- Minimal wx -cut, size 3

In fact, one can get a much stronger result:

Theorem (Menger, 1927)

If u and v are not adjacent, the minimum size of a uv -cut is the maximum number of internally disjoint uv -paths.

Proof (optional): read in the textbook, proof of theorem 4.2.17.
We will see another proof with the Ford-Fulkerson algorithm next week Monday.

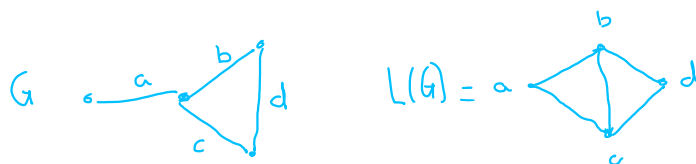
Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Sections 4.1 and 4.2

We progress in our journey to analyzing flow in a network. We first introduce line graphs (and digraphs) to express dual problems, and then move on to networks, flows and capacity.

Line graphs

Goal: Introduce a way to translate edge Menger's theorem and other results on paths in terms of edges.

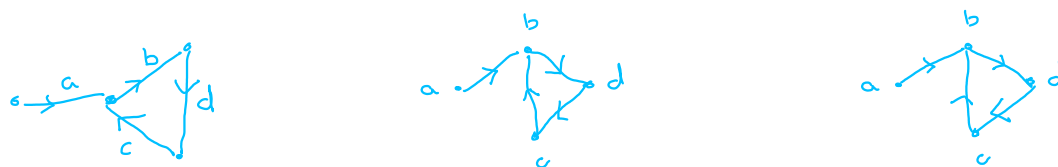
Let $G=(V,E)$ be a graph. Its line graph $L(G)$ has vertices E , and edges of $L(G)$ exist for two edges of G (vertices of $L(G)$) if they are incident to the same vertex in G .



Properties

- The number of edges in $L(G)$ is $\sum_{v \in V} \binom{d_G(v)}{2}$.
- In general, $G \not\cong L(G)$.
- For a graph with no isolated vertex, G is disconnected iff $L(G)$ is.

The same can be done with digraphs. In this case, there is a directed edge from e in E to f if there is a path in $D=(V,E)$ from e to f .



Theorem

If u and v are distinct vertices in a graph (or digraph) G , then the minimum size of an uv -disconnecting set (of edges) equals the maximum size of pairwise edge-disjoint uv -paths.

Sketch of the proof

Use Menger's theorem with $L(G)$.

Deleting an edge in G is equivalent to deleting a vertex in $L(G)$. So the minimum size of a uv -disconnecting set in G is the minimum size of a uv -cut in $L(G)$. By Menger's theorem, this is the maximum number of internally disjoint uv -paths in $L(G)$, which correspond to edge-disjoint paths in G .

Corollary

The edge-connectivity of a graph (or a digraph) is the maximum number k such that there is at least k edge-disjoint uv -paths for all pairs of vertices $\{u, v\}$.

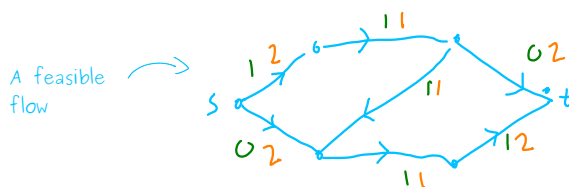
Maximum Network Flow

A network is a directed graph with a nonnegative capacity $c(e)$ on each edge e . A network has distinguished vertices: a source s and a sink t .

A flow f in a network assigns a value $f(e)$ to edge e . For vertices, we write $f^+(v)$ for the total flow of the edges leaving v and $f^-(v)$ for the flow entering v .

A flow is feasible if

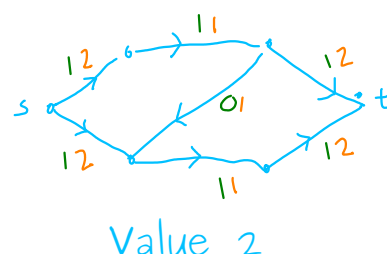
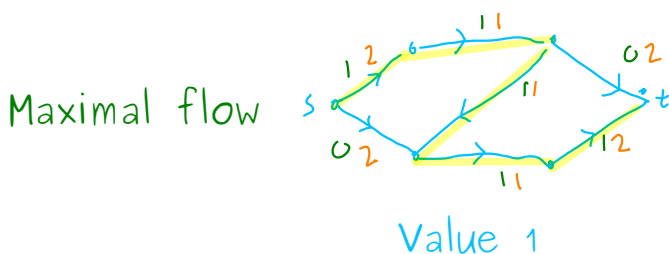
- $0 \leq f(e) \leq c(e)$ for every edge e . Capacity constraint
- $f^+(v) = f^-(v)$ for every vertex except source and sink Conservation constraint



- capacity
- flow

The value of a flow is the net flow of the sink ($f^-(t) - f^+(t)$).

A maximum flow is a feasible flow of maximum value.



Maximum flow

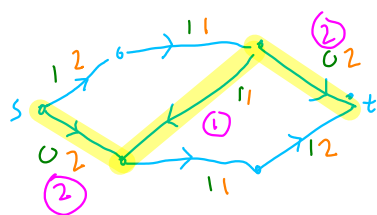
To increase the value of a maximal, but not maximum flow, we use f-augmenting paths. P is an f-augmenting path if

- it is going from source to sink.
- when P follows e in the forward direction, $f(e) < c(e)$.

Let $\varepsilon(e) = c(e) - f(e)$.

- when P follows e in the backward direction, $f(e) > 0$. Let $\varepsilon(e) = f(e)$.

The tolerance of P is the minimum value of $\varepsilon(e)$ over edges in P .



Tolerance 1

Lemma

If P is an f -augmenting path with tolerance z , then we can create a flow f' with value $\text{value}(f) + z$ in the following way:

- if e not in P , $f'(e) = f(e)$
- if e is forward in P , $f'(e) = f(e) + z$
- if e is backward in P , $f'(e) = f(e) - z$.

Proof

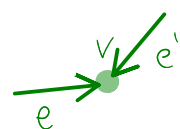
We must prove that f' is a flow (capacity and conservation constraints) and that the result has value z higher than the value of f .

Capacity: If e is forward in P , then the tolerance of e was higher than z , so we can increase its flow by z .

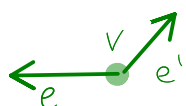
If e is backward in P , then its flow is reduced by z , and it originally was higher than z , so it is still nonnegative.

Conservation: if v is in P (but is neither s nor t), it has either two in-edges (one forward, one backward), two out-edges (also one each way), or one in- and one out-edge (in the same direction).

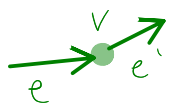
If the two edges are entering v , the flow for the e , the forward edge, is increased by z , and the flow for e' is decreased by z , so the entering flow is unchanged, and the conservation property is maintained.



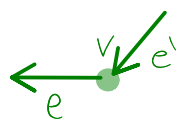
The case of two out-edges is similar, as the exiting flow is decreased by z for the backward edge e , and increased by z for e' .



4



If the flow is forward for the two edges, one is entering and one is leaving, and they are both increased by z . Hence, $f'^+(v) = f^+(v) + z = f^-(v) + z = f'^-(v)$.



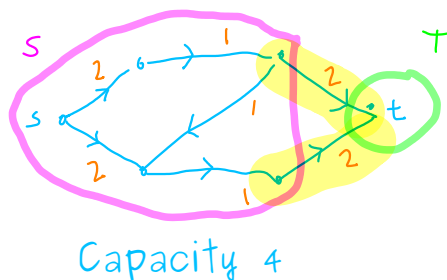
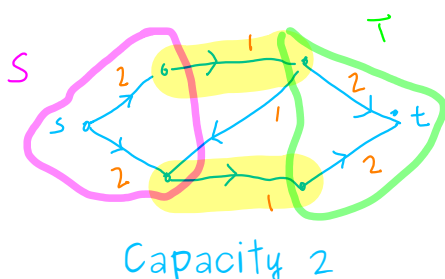
If the flow is backward for both edges, the proof is similar, as the flow is decreased on both edges.

In all cases, the conservation and capacity constraints are satisfied, so the flow is feasible.

The flow is increased by z : P ends at the sink. So P is entering the sink, and $f'^-(t) = f^-(t) + z$ and $f'^+(t) = f^+(t)$. Therefore, $\text{value}(f') = \text{value}(f) + z$. ■

Source/sink cut

Given a partition of the vertices in a network with source s and sink t , consider a partition of the vertices into a source set S (containing s) and a sink set T (with t). A source/sink cut is an edge cut $[S, T]$. Its capacity, $\text{cap}(S, T)$, is the total capacity of the edges from S to T .



Teaser for next class...

Theorem (Max-flow Min-cut, Ford-Fulkerson, 1956)

The maximum flow in a network is the minimum capacity of a source/sink cut.

Recall from last lecture the following theorem (not yet proved):

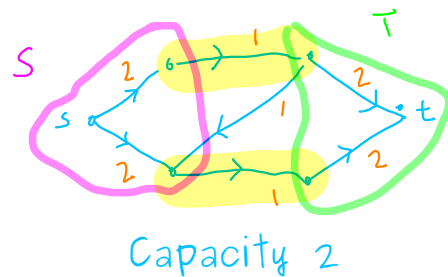
Theorem (Max-flow Min-cut, Ford-Fulkerson, 1956)

The value of a maximum flow in a network is the minimum capacity of a source/sink cut.

Our goal for today is to prove it.

Proposition

Let f be a feasible flow. Any source/sink cut has capacity at least $\text{value}(f)$.



Proof

Consider a source/sink cut $[S, T]$.

Recall that the capacity of the cut is the total capacity of the edges from S to T . For each edge e in $[S, T]$, $f(e) \leq c(e)$.

Also, every path from the source to the sink uses edges from S to T ; otherwise, it contradicts the fact that $[S, T]$ is a cut.

Finally, the value of a flow on a path cannot be higher than the capacity of any edge of the path. Since every path passes through $[S, T]$, every path has maximum capacity that of the edge(s) it uses from the cut.

Hence, the total capacity of the cut is at least the value of the flow.



We have that, for any flow f and any source-sink cut $[S, T]$,

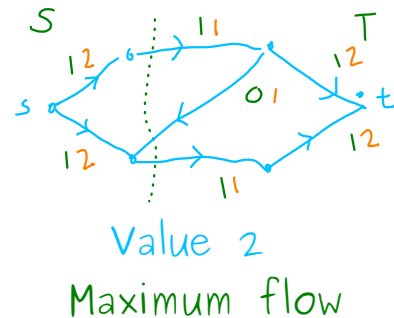
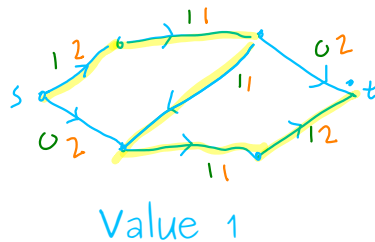
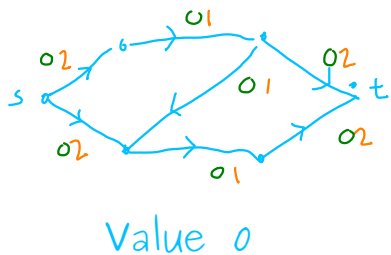
$$\text{value}(f) \leq \text{capacity}([S, T]).$$

That means that if we find a flow with value k and a cut with capacity k in the same network, the flow is necessarily maximum and the cut is necessarily minimal.

Max-flow: Ford-Fulkerson algorithm

Just as in Dijkstra's algorithm, Ford-Fulkerson's algorithm relies on the principle of visiting and reaching vertices. The goal is to find a maximum flow and a minimum cut.

Algorithm to increase the value of a flow, if possible



Input: A feasible flow f (can be the flow that has 0 on every edge)

Output: An f -augmenting path or a cut with capacity $\text{value}(f)$

2 sets of vertices that we update: S (source set) and R (reached).
First, $R = \{s\}$ and S is empty.

Iteration: As long as $R \neq S$, and t is not in R .

Choose v in $R - S$.

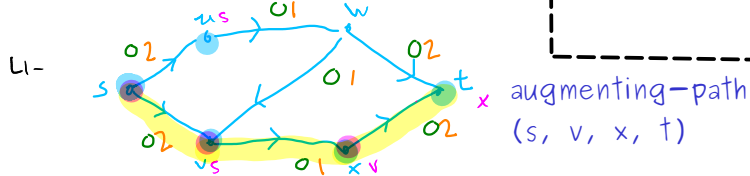
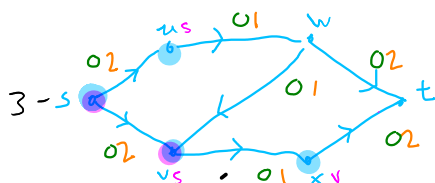
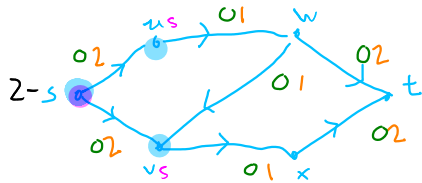
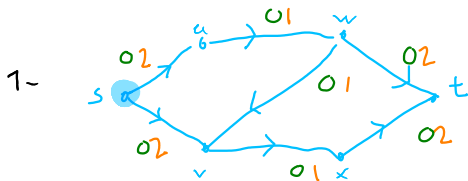
- For each edge vw with $f(vw) < c(vw)$ and $w \notin R$, add w to R .
For w , record that you reached it from v .
- For each edge uv with nonzero $f(uv)$ and $u \notin R$, add u to R .
For u , record that you reached it from v .

After visiting all the edges incident to v , add v to S .

If t is reached, we found an augmenting path. Return it (using the recorded vertices). If $R = S$, there is no possibility of f -augmenting path, so $[S, V - S]$ is a source-sink cut.

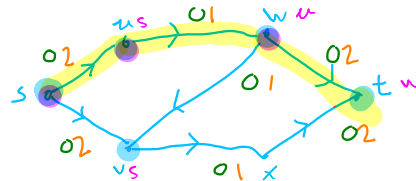
To get a maximum flow:

start with the zero flow, and use the former algorithm to find an augmenting-path. When you get a cut, the flow is maximal.



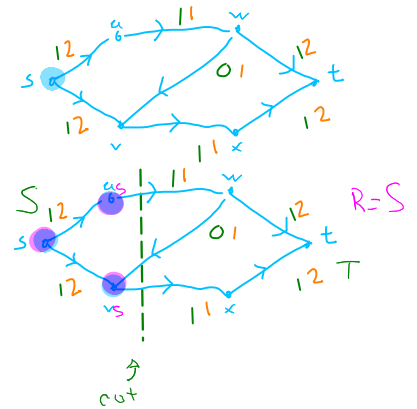
- Flow
- Capacity
- Visited vertices (S set)
- Reached vertices (R set)

Or alternatively



augmenting-path
(s, u, w, t)

Starting with a different flow:



Max-Flow Min-Cut Theorem

Theorem (Max-flow Min-cut, Ford-Fulkerson, 1956)

The value of a maximum flow in a network is the minimum capacity of a source/sink cut.

Sketch of proof

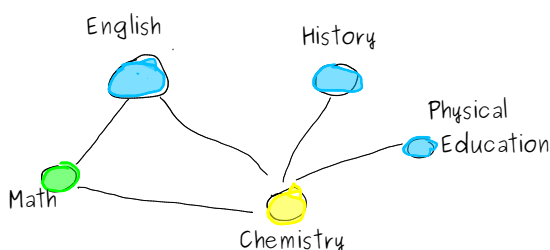
The key is to use the algorithm, starting with any feasible flow.

- If it terminates with t, we found an augmenting path: all the edges going forward are not used at full capacity, and all the edges going backward have nonzero flow. This means we increase the flow by the tolerance of the augmenting path.
- If it terminates with a cut, the flow is maximal because the edges leaving S are at maximal capacity.

Recall from the very first lecture the following problem:

Scheduling and avoiding conflicts

My high school used to have a very long exam sessions at the end of the year, and there were still some conflicts. I wish the administrators knew graph theory...



Vertices: Subjects

Edges: If someone takes both subjects, i.e. eventual scheduling conflicts.

Intuitive definition:

A proper coloring of a graph is a partition of the vertices into independent sets.

Scheduling with no conflicts is equivalent to coloring.

If we want to use the minimum time, we should use as few colors as possible.

Definition

A k-coloring of a graph G is a labeling of the vertices using labels from a set of size k (called colors, even though the labels can be numbers, for example).

The vertices of one color form a color class.

A coloring is proper if no two adjacent vertices have the same label.

A graph is k-colorable if it has a proper k -coloring.

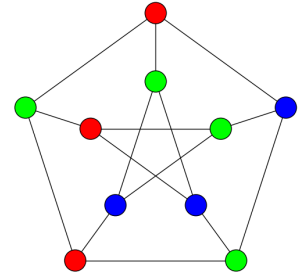
The chromatic number $\chi(G)$ is the least k such that G is k -colorable.

In a proper coloring, every color class is an independent set. The chromatic number is the smallest number of independent sets that partition the vertices of a graph.

Example

The Petersen graph has chromatic number 3:

- It is not 2-colorable, because its vertices cannot be divided into two independent sets; it would otherwise be bipartite.
- It is 3-colorable, as shown on the right.



Notice that the chromatic number is an extremal problem: we need to show it is minimal and that a proper coloring exists.

Colorings for non-simple graphs

Graphs with loops do not admit proper colorings: a vertex that is incident to a loop could not be colored.

Every loopless graph can be colored: a trivial coloring where every vertex has a distinct color would work.

Multiple edges don't change anything to colorings, as two adjacent vertices cannot be colored the same color regardless of the number of edges between them.

Optimality

A graph G is k -chromatic if $k = \chi(G)$; a proper k -coloring is then an optimal coloring.

If $\chi(H) < \chi(G) = k$ for every subgraph H of G , then G is k -critical or color-critical.

Examples

$k=1$ 

$k=3$: The 3-critical graphs are the smallest graphs that are not bipartite: these are the odd cycles.

$k=2$ 



Not 3-critical

No general characterization of 4-critical is known.

3

First bounds on the chromatic number

The clique number of a graph, written $\omega(G)$, is the maximum size of a clique in G . (Recall that a clique is a complete subgraph).

Also, recall that the independence number, $\alpha(G)$, is the size of a maximum independent set.

Proposition

For every graph $G=(V,E)$, $\chi(G) \geq \omega(G)$ and $\chi(G) \geq |V|/\alpha(G)$.

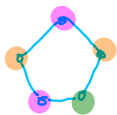
Proof

If there is a clique of size k , the k vertices in the clique must be of different colors.

For the second inequality, rewrite it as $\chi(G)\alpha(G) \geq |V|$. $\chi(G)$ is the number of color classes, and $\alpha(G)$ is the maximum size of a color class.



The chromatic number is not necessarily the size of the maximal clique:



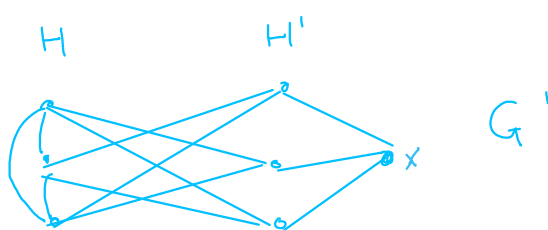
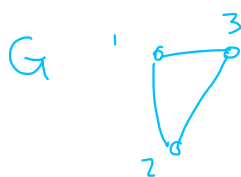
Maximal clique has size 2

Chromatic number is 3

Example: Mycielski's construction

From a simple graph G , construct a graph G' in the following way:

Let H and H' be two copies of G , but delete all edges from H' . If vertices u and v are adjacent in G , draw an edge between u in H and v' in H' (the copy of v in H'). Add an extra vertex x and connect it to all the vertices in H' .



Notice that u and u' are never adjacent.

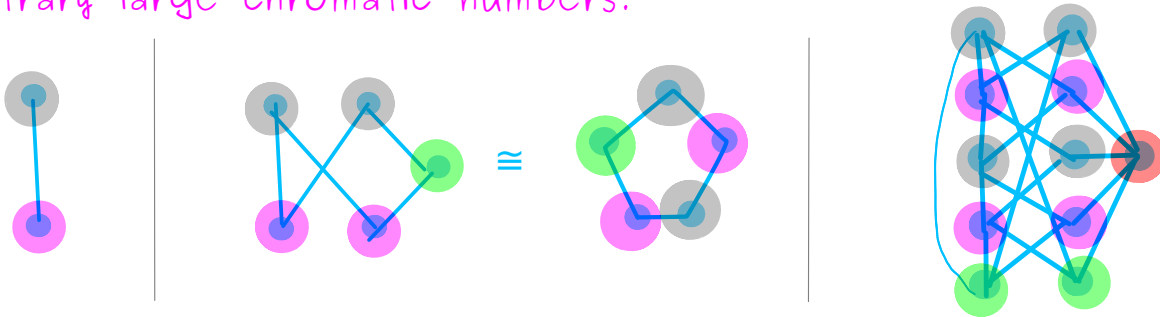
If G has chromatic number k , then G' has chromatic number $k+1$:

The colors in H and in H' can be the same. In G , u and v can have the same color if they are not adjacent. Hence, u and v' (as well as v and u') are not adjacent in G' , so they can have the same color. Hence x is the only vertex with a new color added.

So graphs obtained by iterating this process can have arbitrarily large chromatic number.

Question: What is the clique number of a graph obtained with the Mycielski's construction?

Mycielski's construction is used to build triangle-free graphs with arbitrary large chromatic numbers:

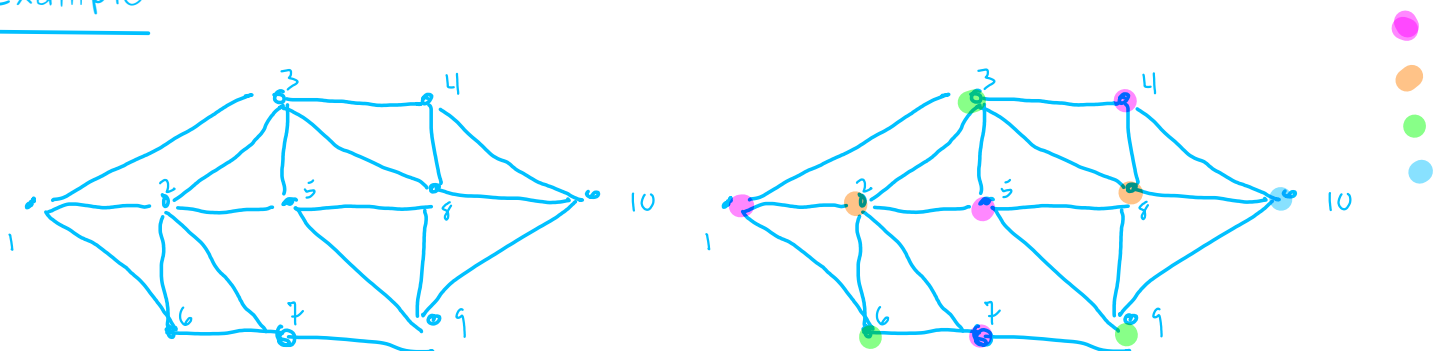


If we start from a triangle-free graph, the Mycielski construction is triangle-free.

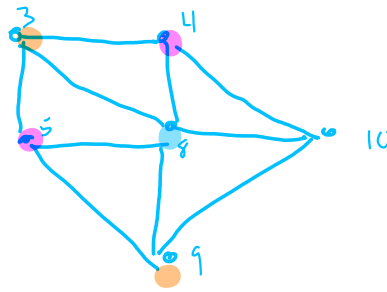
Greedy coloring algorithm

- Order the vertices $\{1, 2, \dots, n\}$. We will color the vertices using numbers $\{1, 2, \dots, n\}$.
- For every vertex (in order), label it with the smallest color not already in use in its neighborhood.

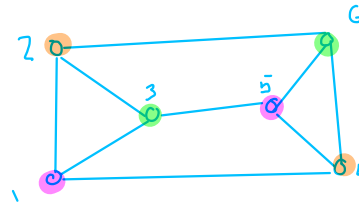
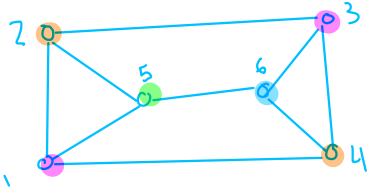
Example



In this case, it is actually minimal. This graph cannot be colored with fewer than 4 colors.



The coloring does not always use the minimum number of colors:



Proposition

The chromatic number is at most $\Delta(G)+1$.

Proof

The greedy algorithm described above yields a proper coloring. In the worst case, all neighbors of one vertex have distinct color, and we must add a color. When this happens, the number of colors is one more than the number of neighbors; that is at most $\Delta(G)+1$.



Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001. Section 5.1.

Last class, I introduced proper colorings of graphs, and the chromatic number. We also looked at some bounds on the chromatic number, and we keep exploring bounds on the chromatic number today.

So far, we know:

- The chromatic number can be bounded in terms of the independence number and the clique number: $\chi(G) \geq \omega(G)$ and $\chi(G) \geq |V|/\alpha(G)$.
- The chromatic number can be bounded in terms of the maximum degree: $\chi(G) \leq \Delta(G) + 1$.

These bounds are easy to check, but they are not the best possible.

Another upper bound

Theorem (Brooks, 1941)

If G is connected, and is not the complete graph nor an odd cycle, $\chi(G) \leq \Delta(G)$.

Examples and special cases

If $\Delta(G)=0$, then G has 1 vertex (because it is connected), and is thus the complete graph. So no graph in this case satisfies the hypotheses of the theorem.

If $\Delta(G)=1$, then G has 2 vertices, and this is again the complete graph.

If $\Delta(G)=2$, G is either a cycle or a path. Open paths and even cycles are bipartite, so their chromatic number is 2, which also is the maximum degree. Odd cycles are excluded from by hypothesis of the theorem.

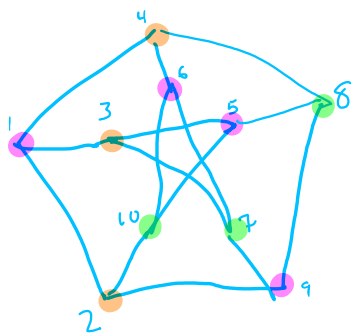
Complete graphs don't satisfy the inequality, as their chromatic number is one more than the maximum degree (every vertex must have different colors).

The hypothesis that the graph is connected is needed to avoid the case of having only isolated vertices. ②

- Not complete, maximum degree is 0.
- Chromatic number is 1.

Notice that, whenever a graph with n vertices is not the complete graph, the chromatic number is at most $n-1$: Since there is at least one pair of non-adjacent vertices in a non-complete graph, they can be the same colors. So n colors are never needed if the graph is not complete.

Example: Coloring the Petersen graph using the greedy algorithm



The Petersen graph is 3-regular.

It satisfies the hypothesis of the theorem, so it must have maximum degree 3. That means there exists an ordering of the vertices that allows it.

Proof of Brooks' Theorem

We already inspected the case where the largest degree is at most 2, so assume $\Delta(G)=k$ is at least 3.

If G is not k -regular:

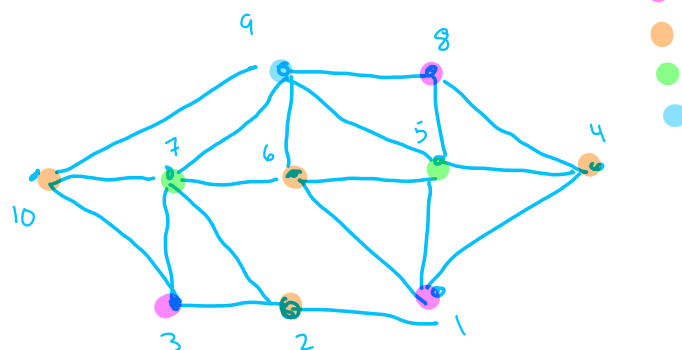
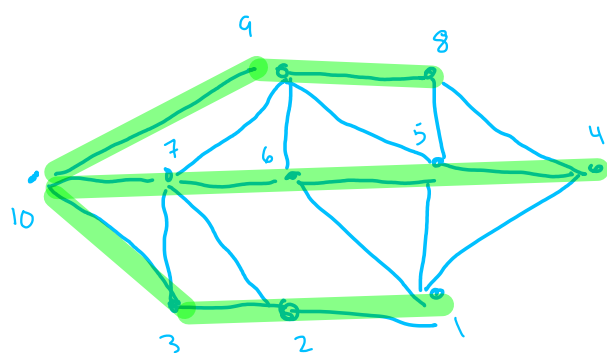
Then, there is a vertex v with degree less than k . Let T be a spanning tree in G (which is possible since the graph is connected).

We will use this spanning tree for ordering the vertices. The goal is to find the right ordering for the vertices, and then apply the greedy algorithm from last lecture.

- Number vertex v with n (last vertex to be colored).
- Label the other vertices in decreasing order on paths leaving v in T .
- Color the vertices using the greedy algorithm from last lecture.

Every time we color a new vertex u (that is not v), there are at most $k-1$ of its neighbors that have been previously colored, so k colors are enough.

For the last step, we know that v has at most $k-1$ neighbors, so in the worst case, a k -th color will be necessary to color it. In total, k colors are enough if the graph is not k -regular.



A similar process holds if the graph is k -regular, but there are two cases:

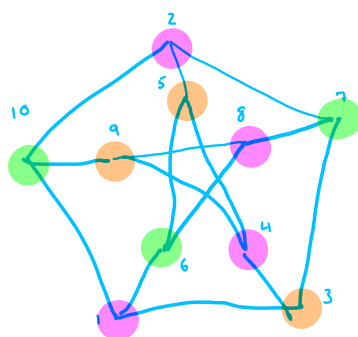
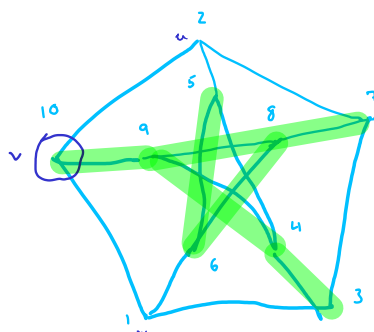
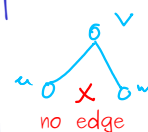
- There is a cut-vertex v . Then, $G - \{v\}$ is disconnected, and each component can be colored with k colors. Place the colors in the components so that vertices incident to v have the same color in both components.

Then, v can be colored using any other color, so G is k -colorable.



- There is no cut-vertex, meaning that G is 2-connected.

If G has a vertex v with two neighbors that are not adjacent u and w such that $G - \{u, w\}$ is connected, we can use a similar argument. We label u and w by 1 and 2, and create a spanning tree in $G - \{u, w\}$. Starting from v , we label the vertices in decreasing order and obtain a proper k -coloring of G because the last vertex has two vertices (u and w) colored the same.



I claim there is always such a triple of vertices when G is 2-connected and k -regular, with $k \geq 3$. (The details of this are in the textbook.) ■

Subgraph, cliques and chromatic number

Proposition

If H is a subgraph of G , $\chi(H) \leq \chi(G)$.

Proof

All the edges of H are in G , so the vertices of G cannot be colored with fewer than $\chi(H)$ vertices (however, if we add edges, they might need more colors). ■

This is similar to the proposition we had in last lecture: $\chi(G) \geq \omega(G)$. However, cliques are not needed to have large chromatic number (as exhibited by the graphs build using Mycielski's construction).

Proposition

Every k -chromatic graph has at least $\binom{k}{2}$ edges.

Proof

Consider an optimal coloring of the graph. Since it uses the minimum number of colors, there is at least one edge connecting two color classes; otherwise, there are two classes (blue and red) with no edges between the two classes, and all the red vertices can be colored blue. Hence, we need at least one edge per pair of colors, that is $\binom{k}{2}$ edges.

This is achieved by complete graphs K_k . ■

Reference: Douglas B. West. Introduction to graph theory, 2nd edition, 2001.
Sections 5.1 and 5.2