Graphs

Recall: a *graph* G is a pair (V, E), where V is a set of *vertices* or *nodes* and E is a set of *edges*

- We sometimes write $G(V, E)$ instead of G
- We sometimes write $V(G)$ and $E(G)$ to denote the vertices/edges of G.

In a *directed graph (digraph)*, the edges are ordered pairs $(u, v) \in V \times V$:

- the edge goes from node u to node v

In an *undirected simple graphs*, the edges are sets $\{u, v\}$ consisting of two nodes.

- there's no direction in the edge from u to v
- More general (non-simple) undirected graphs (sometimes called multigraphs) allow self-loops and multiple edges between two nodes.
  - There may be several nodes between two towns

Directed vs. Undirected Graphs

Is the following better represented as (a) a directed graph or (b) an undirected graph:

1. Social network (edge between u and v if u and v are friends)

Directed vs. Undirected Graphs

Is the following better represented as (a) a directed graph or (b) an undirected graph:

1. Social network (edge between u and v if u and v are friends)
2. Niche graph (edge between species u and v if they compete)

Directed vs. Undirected Graphs

Is the following better represented as (a) a directed graph or (b) an undirected graph:

1. Social network (edge between u and v if u and v are friends)
2. Niche graph (edge between species u and v if they compete)
3. influence graph (edge between u and v if u influences v)

Directed vs. Undirected Graphs

Is the following better represented as (a) a directed graph or (b) an undirected graph:

1. Social network (edge between u and v if u and v are friends)
2. Niche graph (edge between species u and v if they compete)
3. influence graph (edge between u and v if u influences v)
4. communication network (edge between u and v if u and v are connected by a communication link)

Degree

In a directed graph $G(V, E)$, the *indegree* of a vertex v is the number of edges coming into it

- $\text{indegree}(v) = |\{v' : (v', v) \in E\}|$

The *outdegree* of v is the number of edges going out of it:

- $\text{outdegree}(v) = |\{v' : (v, v') \in E\}|$

The *degree* of v, denoted $\deg(v)$, is the sum of the indegree and outdegree.

For an undirected simple graph, it doesn't make sense to talk about indegree and outdegree. The degree of a vertex is the sum of the edges incident to the vertex.

- edge e is *incident to* v if v is one of the endpoints of e

Theorem: Given a graph $G(V, E)$,

$$2|E| = \sum_{v \in V} \deg(v)$$

Proof: For a directed graph: each edge contributes once to the indegree of some vertex, and once to the outdegree of some vertex. Thus $|E|$ = sum of the indegrees = sum of the outdegrees.

Same argument for a simple undirected graph.

- This also works for a general undirected graph, if you double-count self-loops

Handshaking Theorem

Theorem: The number of people who shake hands with an odd number of people at a party must be even.

Proof: Construct a graph, whose vertices are people at the party, with an edge between two people if they shake hands. The number of people person p shakes hands with is $\deg(p)$. Split the set of all people at the party into two subsets:

- A = those that shake hands with an even number of people
- B= those that shake hands with an odd number of people

Handshaking Theorem

Theorem: The number of people who shake hands with an odd number of people at a party must be even.

Proof: Construct a graph, whose vertices are people at the party, with an edge between two people if they shake hands. The number of people person p shakes hands with is $\deg(p)$. Split the set of all people at the party into two subsets:

- A = those that shake hands with an even number of people
- B= those that shake hands with an odd number of people

$$\sum_{p} \deg(p) = \sum_{p \in A} \deg(p) + \sum_{p \in B} \deg(p)$$

- We know that $\sum_p \deg(p) = 2|E|$ is even.
- $\sum_{p \in A} \deg(p)$ is even, because for each $p \in A$, $\deg(p)$ is even.
- Therefore, $\sum_{p \in B} \deg(p)$ is even.
- Therefore $|B|$ is even (because for each $p \in B$, $\deg(p)$ is odd, and if $|B|$ were odd, then $\sum_{p \in B} \deg(p)$ would be odd).

Graph Isomorphism

When are two graphs that may look different when they're drawn, really the same?

Answer: $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ are *isomorphic* if they have the same number of vertices ($|V_1| = |V_2|$) and we can relabel the vertices in G_2 so that the edge sets are identical.

- Formally, G_1 is isomorphic to G_2 iff (there is a bijection $f : V_1 \to V_2$ such that $\{v, v'\} \in E_1$ iff $(\{f(v), f(v')\} \in E_2)$.
- Note this means that $|E_1| = |E_2|$

Checking for Graph Isomorphism

There are some obvious requirements for $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ to be isomorphic:

- $|V_1| = |V_2|$
- $|E_1| = |E_2|$
- for each d, #(vertices in V_1 with degree d) = #(vertices in V_2 with degree d)

Checking for isomorphism is in NP:

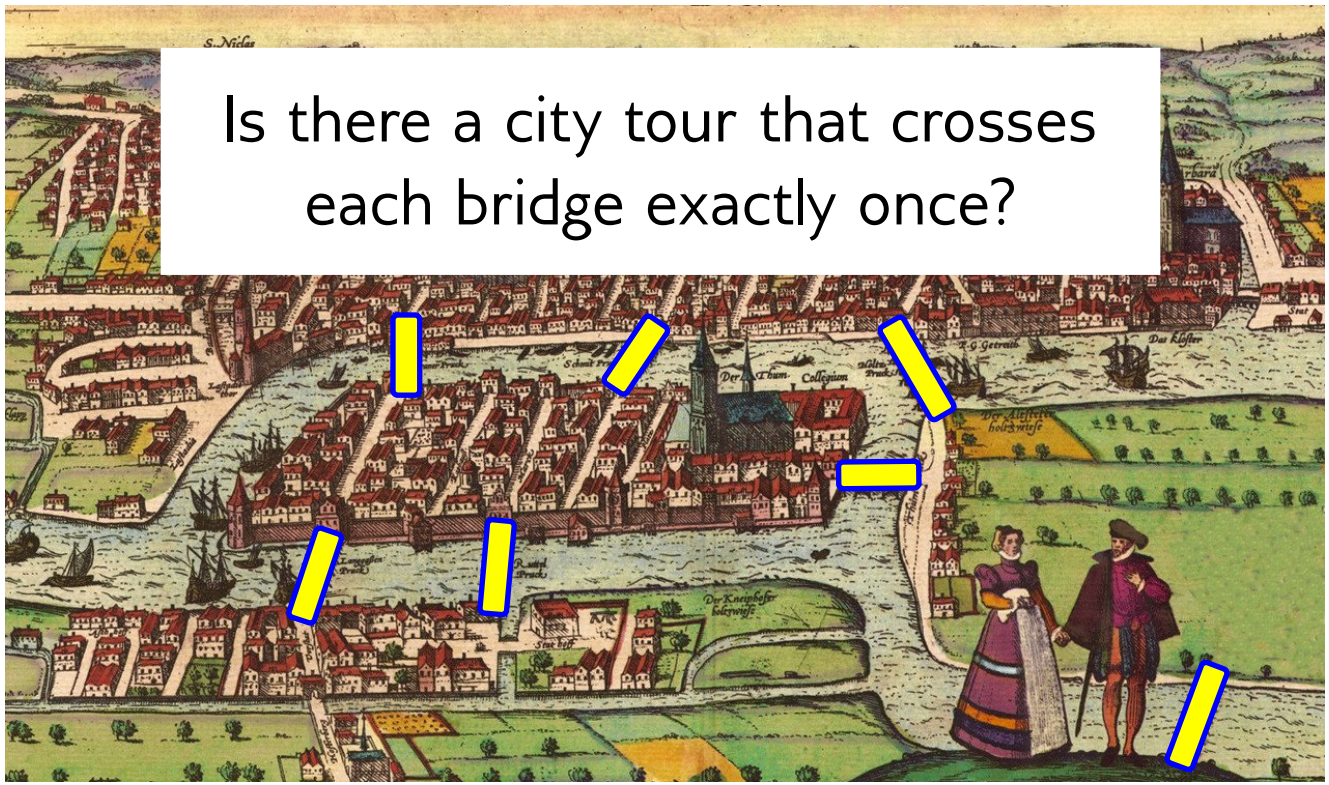
- Guess an isomorphism f and verify
- We believe it's not in polynomial time and not NP complete.

Paths

Given a graph $G(V, E)$.

- A *path* in G is a sequence of vertices $(v_0, \ldots, v_n)$ such that $\{v_i, v_{i+1}\} \in E$ ((v_i, v_{i+1}) in the directed case).
  - a simple path has no repeated vertices
    - MCS calls a path a *walk* and a simple path a *path*
  - vertex v is *reachable* from u if there is a path from u to v
- If $v_0 = v_n$, the path is a *cycle*
- An *Eulerian* path/cycle is a path/cycle that traverses every every edge in E exactly once
- A *Hamiltonian* path/cycle is a path/cycle that passes through each vertex in V exactly once.
- A graph with no cycles is said to be *acyclic*
- An undirected graph is *connected* if every vertex is reachable from every other vertex.

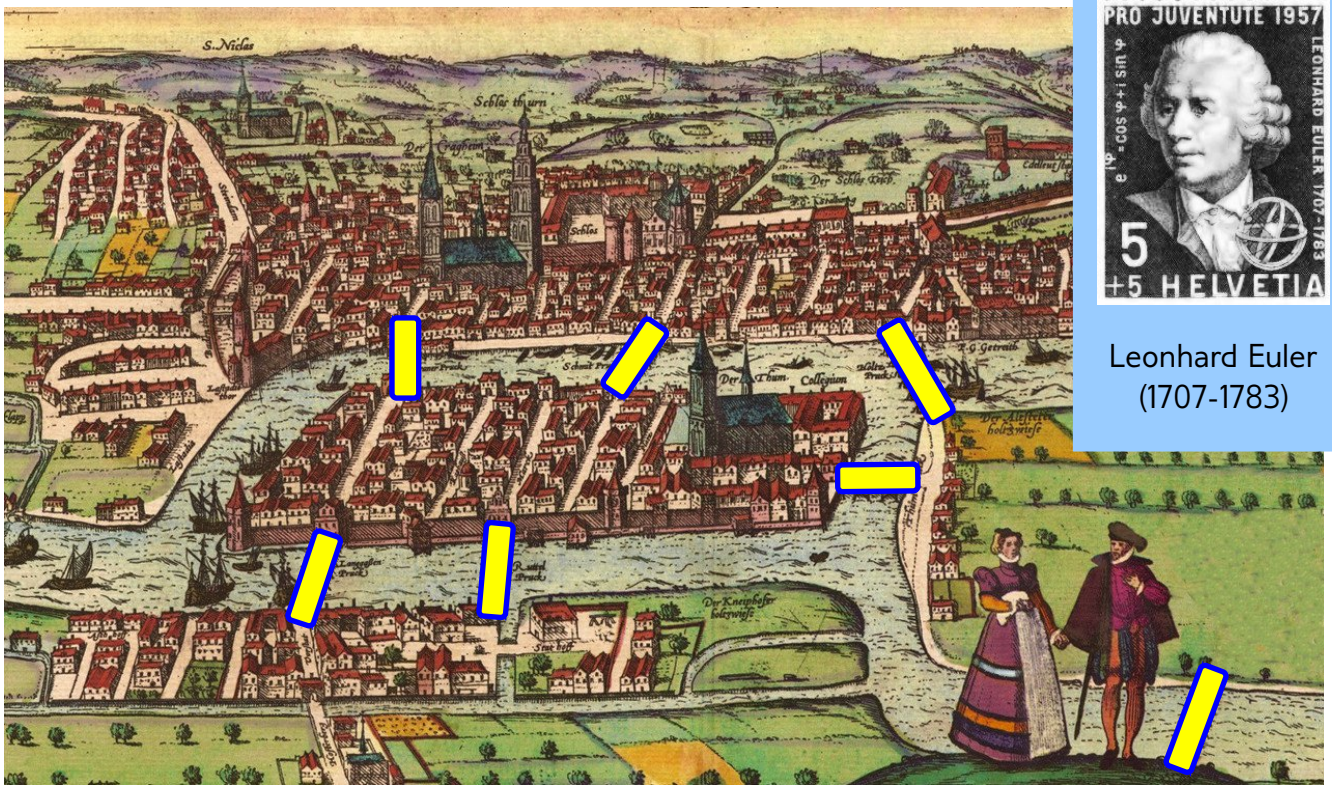
Bridges of Königsberg



Braun & Hogenberg, "Civitates Orbis Terrarum", Cologne 1585. Photoshopped to clean up right side and add 7th bridge.

Remember this from the first class?

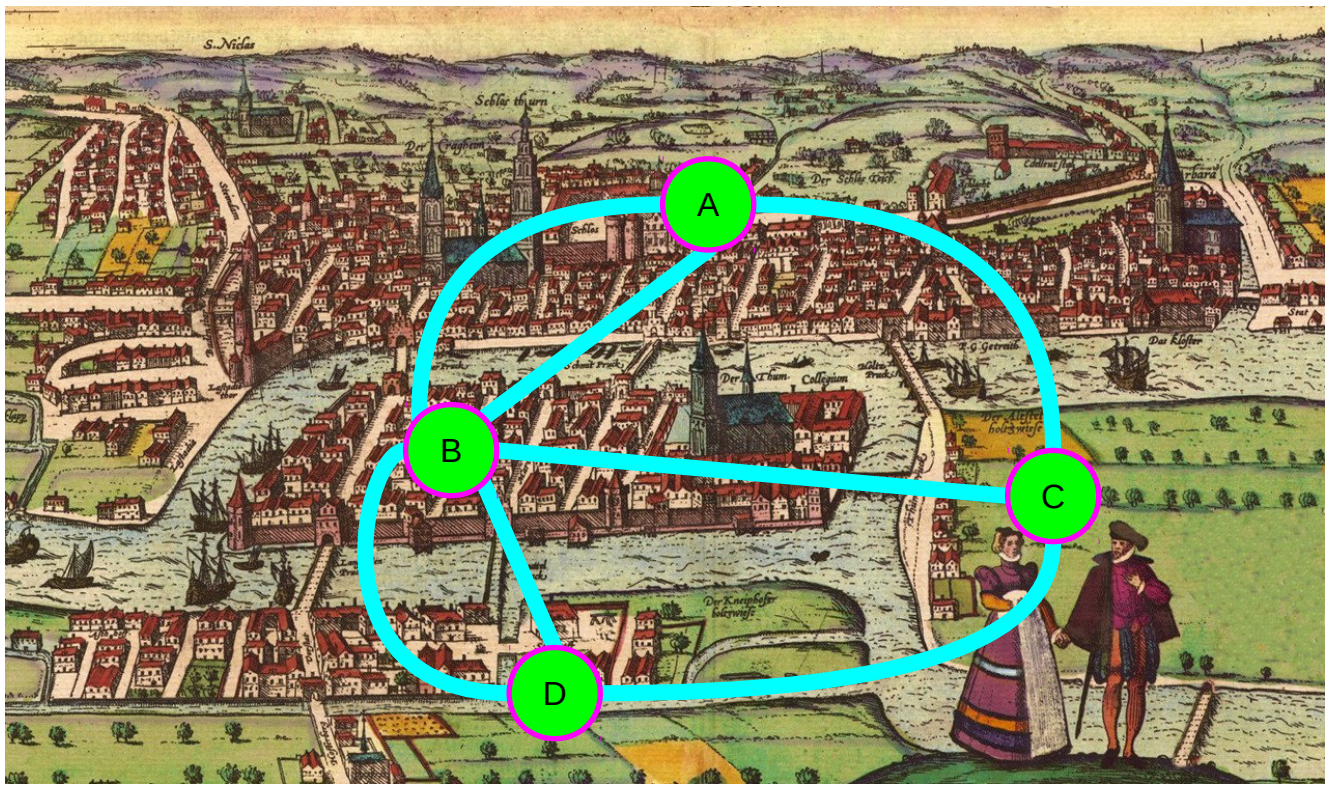
Bridges of Königsberg



Leonhard Euler
(1707-1783)

Braun & Hogenberg, “Civitates Orbis Terrarum”, Cologne 1585. Photoshopped to clean up right side and add 7th bridge.

Bridges of Königsberg



Braun & Hogenberg, "Civitates Orbis Terrarum", Cologne 1585. Photoshopped to clean up right side and add 7th bridge.

Euler's key insight: represent the problem as a graph

Eulerian Paths

Recall that $G(V, E)$ has an Eulerian path if it has a path that goes through every edge exactly once. It has an Eulerian cycle (or Eulerian circuit) if it has an Eulerian path that starts and ends at the same vertex.

How can we tell if a graph has an Eulerian path/circuit?

What's a necessary condition for a graph to have an Eulerian circuit?
Count the edges going into and out of each vertex:

- Each vertex must have even degree!

This condition turns out to be sufficient too.

Eulerian Paths

Recall that $G(V, E)$ has an Eulerian path if it has a path that goes through every edge exactly once. It has an Eulerian cycle (or Eulerian circuit) if it has an Eulerian path that starts and ends at the same vertex.

How can we tell if a graph has an Eulerian path/circuit?

What's a necessary condition for a graph to have an Eulerian circuit?
Count the edges going into and out of each vertex:

- Each vertex must have even degree!

This condition turns out to be sufficient too.

Theorem: A connected (multi)graph has an Eulerian cycle iff each vertex has even degree.

Finding Cycles

Proof: The necessity is clear: In the Eulerian cycle, there must be an even number of edges that start or end with any vertex.

Finding Cycles

Proof: The necessity is clear: In the Eulerian cycle, there must be an even number of edges that start or end with any vertex.

Lemma 1: Every graph where every vertex has even degree can be written as an edge-disjoint union of cycles.

Proof: By induction on the number of edges. Follow your nose, walking on the graph, never using the same edge twice, until you can't leave a vertex (because you've already used all its edges). At this point, you must have a cycle: Suppose not. Then you walked on a path $P = v_0 - v_1 - \cdots - v_m$, If $v_0 \neq v_m$, then in the graph G' that results from removing the edges on P, v_0 and v_m have odd degree, and all other vertices have even degree (Proof: by induction on the length of the path). Since v_m has odd degree in G', there must be an edge going out of v_m that you haven't used yet – a contradiction! That means we must eventually get a cycle.

Finding Cycles

Proof: The necessity is clear: In the Eulerian cycle, there must be an even number of edges that start or end with any vertex.

Lemma 1: Every graph where every vertex has even degree can be written as an edge-disjoint union of cycles.

Proof: By induction on the number of edges. Follow your nose, walking on the graph, never using the same edge twice, until you can't leave a vertex (because you've already used all its edges). At this point, you must have a cycle: Suppose not. Then you walked on a path $P = v_0 - v_1 - \cdots - v_m$, If $v_0 \neq v_m$, then in the graph G' that results from removing the edges on P, v_0 and v_m have odd degree, and all other vertices have even degree (Proof: by induction on the length of the path). Since v_m has odd degree in G', there must be an edge going out of v_m that you haven't used yet – a contradiction! That means we must eventually get a cycle.

Once we have found a cycle, we can remove the edges on this cycle from G to get G'. Every edge in G' still has even degree, so G' is the union of edge-disjoint cycles. That means G is too.

Finding the Eulerian cycle

Let $C_1, \ldots, C_k$ be the cycles that make up G. Here's one way of getting an Eulerian cycle (I gave a simpler approach afterwards):

- Start walking on $C_1 = C_{i_1}$ until you either come back to the beginning (it's a cycle) or you reach a vertex that is on some other path $C_{i_2} \neq C_{i_1}$. In the latter case, start walking on C_{i_2} until you either come back to the beginning, in which case you continue walking on C_{i_1}, or you reach a vertex that is on $C_{i_3} \notin \{C_{i_1}, C_{i_2}\}$. In the latter case, start walking on C_{i_3} until ....
  - Formally, the procedure is recursive. You should be able to write a program that does this.

Claim. you eventually walk through all the cycles exactly once, so you've walked along an Eulerian path.

How do we prove this?

Lemma 2: Suppose that this process reaches cycles $D_1, \ldots, D_m$. Then we walk along every edge in $D_1 \cup \ldots \cup D_m$.

Proof: By induction on m.

Lemma 2: Suppose that this process reaches cycles $D_1, \ldots, D_m$. Then we walk along every edge in $D_1 \cup \ldots \cup D_m$.

Proof: By induction on m.

Lemma 3: Every cycle is reached.

Proof: Suppose C is not reached. Let v be a vertex in C_1 and v' be a vertex in C. Since the graph is connected, there is a path $v_0 \ldots v_m$, where $v = v_0$ and $v' = v_m$ Consider the first vertex v_j on this path that is not reached. But it should be (since we should have detoured to do the cycle that includes (v_{j-1}, v_j).

Lemma 2: Suppose that this process reaches cycles $D_1, \ldots, D_m$. Then we walk along every edge in $D_1 \cup \ldots \cup D_m$.

Proof: By induction on m.

Lemma 3: Every cycle is reached.

Proof: Suppose C is not reached. Let v be a vertex in C_1 and v' be a vertex in C. Since the graph is connected, there is a path $v_0 \ldots v_m$, where $v = v_0$ and $v' = v_m$ Consider the first vertex v_j on this path that is not reached. But it should be (since we should have detoured to do the cycle that includes (v_{j-1}, v_j).

Theorem: If G has exactly two vertices v and v' of odd degree, then G has an Eulerian path starting at v and ending at v'.

Lemma 2: Suppose that this process reaches cycles $D_1, \ldots, D_m$. Then we walk along every edge in $D_1 \cup \ldots \cup D_m$.

Proof: By induction on m.

Lemma 3: Every cycle is reached.

Proof: Suppose C is not reached. Let v be a vertex in C_1 and v' be a vertex in C. Since the graph is connected, there is a path $v_0 \ldots v_m$, where $v = v_0$ and $v' = v_m$ Consider the first vertex v_j on this path that is not reached. But it should be (since we should have detoured to do the cycle that includes (v_{j-1}, v_j).

Theorem: If G has exactly two vertices v and v' of odd degree, then G has an Eulerian path starting at v and ending at v'.

Proof: Just add an edge connecting v and v' to get G^+. All vertices in G have even degree, so it has an Eulerian cycle. Then remove the edge from v to v' to get an Eulerian path.

A simpler approach

We proceed by (strong) induction on the number of edges in the graph G to show that any connected graph where all vertices have even degree has an Eulerian cycle. Clearly there is an Eulerian path if G has 0 edges.

A simpler approach

We proceed by (strong) induction on the number of edges in the graph G to show that any connected graph where all vertices have even degree has an Eulerian cycle. Clearly there is an Eulerian path if G has 0 edges.

So suppose that G has $n+1$ edges. Compute a cycle C using the algorithm of Lemma 1. Remove the edges of the cycle to get a graph $G' = (V, E')$. It's easy to check that each vertex of G' still has even degree. Now G' may be disconnected. Suppose that the connected components are $G_1, \ldots, G_m$. By induction, each of $G_1, \ldots, G_m$ has an Eulerian cycle, call them $C_1, \ldots, C_m$. Moreover, each of these cycles must have a vertex in common with C (since G was connected, and removing the edges in C disconnected G.

Now we can get an Eulerian cycle for G as follows: Start walking along C until you reach a vertex on one of $C_1, \ldots, C_m$, say C_j. When you do, walk around C_j. Then continue walking on C until you reach a vertex on a cycle that you haven't seen before, say $C_{j'}$. Walk around $C_{j'}$. Then continue walking along C until you reach the next cycle that you have seen before. Clearly, this gives a Euclidean cycle for G.

Now we can get an Eulerian cycle for G as follows: Start walking along C until you reach a vertex on one of $C_1, \ldots, C_m$, say C_j. When you do, walk around C_j. Then continue walking on C until you reach a vertex on a cycle that you haven't seen before, say $C_{j'}$. Walk around $C_{j'}$. Then continue walking along C until you reach the next cycle that you have seen before. Clearly, this gives a Euclidean cycle for G.

Next we give an algorithm that implements this:

An Algorithm for Finding Eulerian Paths

Step 1: Find a cycle using the following procedure:

Input: $G(V, E)$ [a list of vertices and edges]
procedure Pathgrow(V,E,v)
[v is first vertex in cycle]
$P \leftarrow ()$ [P is sequence of edges on cycle]
$w \leftarrow v$ [w is last vertex in P]
repeat until $I(w) - P = \emptyset$
[$I(w)$ is the set of edges incident on w]
Pick $e \in I(w) - P$
$w \leftarrow$ other end of e
$P \leftarrow P \cdot e$ [append e to P]
return P

An Algorithm for Finding Eulerian Paths

Step 1: Find a cycle using the following procedure:

Input: $G(V, E)$ [a list of vertices and edges]
procedure Pathgrow(V,E,v)
[v is first vertex in cycle]
$P \leftarrow ()$ [P is sequence of edges on cycle]
$w \leftarrow v$ [w is last vertex in P]
repeat until $I(w) - P = \emptyset$
[$I(w)$ is the set of edges incident on w]
Pick $e \in I(w) - P$
$w \leftarrow$ other end of e
$P \leftarrow P \cdot e$ [append e to P]
return P

Claim: If every vertex in V has even degree, then P will be a cycle

- Loop invariant: In the graph $G(V, E - P)$, if the first vertex (v) and last vertex (w) in P are different, they have odd degree; all the other vertices have even degree.

Now put the cycles together:

procedure Euler(V,E,v)
    //$G(V, E)$ is a connected undirected graph//
    //$v \in V$ is arbitrary//
    //output is an Eulerian cycle in G//
Pathgrow(V',E',v') [returns cycle P in G]
if P is not Eulerian
then delete the edges in P from E;
    let $G_1(V_1, E_1), \ldots, G_n(V_n, E_n)$ be
        the resulting connected components
    let v_i be a vertex in V_i also on P
    for $i = 1$ **to** n
        Euler(V_i, E_i, v_i) [returns Eulerian cycle C_i]
        Attach C_i to P at v_i
    endfor
return P

Hamiltonian Paths

Recall that $G(V, E)$ has a Hamiltonian path if it has a path that goes through every vertex exactly once. It has a Hamiltonian cycle (or Hamiltonian circuit) if it has a Hamiltonian path that starts and ends at the same vertex.

There is no known easy characterization or algorithm to check if a graph has a Hamiltonian cycle/path.

Graphs and Scheduling

In a scheduling problem, there are tasks and constraints specifying that some tasks have to be completed before others.

- This can be represented graphically
- The nodes are the tasks
- There is an edge from u to v if u has to be completed before v is started
- Sometimes the edges have *weights* (how much time it takes to complete the task)

The result is a *directed acyclic graph (dag)*.

- If there's a cycle, the job can't be completed!

This way of representing task scheduling gives us lots of useful information:

- the length of the longest path in the graph gives us a lower bound on how long the task takes (even if we could throw an unbounded number of people at the tasks, and they can work in parallel).
- It can take longer if we have a bounded number of workers

Topological Sorts

If there's only one person doing the tasks, in what order should we do the tasks in a scheduling problem?

A *topological sort* of a dag $G = (V, E)$ is a list of all the vertices in V such that if v is reachable from u, then u precedes v on the list.

- A topological sort of a dag representing tasks gives a possible order to do the tasks
- There may be more than one topological sort of a dag

Theorem: Every dag $G = (V, E)$ has a topological sort.

How can we prove this formally?

Topological Sorts

If there's only one person doing the tasks, in what order should we do the tasks in a scheduling problem?

A *topological sort* of a dag $G = (V, E)$ is a list of all the vertices in V such that if v is reachable from u, then u precedes v on the list.

- A topological sort of a dag representing tasks gives a possible order to do the tasks
- There may be more than one topological sort of a dag

Theorem: Every dag $G = (V, E)$ has a topological sort.

How can we prove this formally?

- By induction on the number of vertices in V
  - Remove a node that has no edges going out of it.
  - How do you know there is one?

Graph Coloring

How many colors do you need to color the vertices of a graph so that no two adjacent vertices have the same color?

- Application: scheduling
  - Vertices of the graph are courses
  - Two courses taught by same prof are joined by edge
  - Colors are possible times class can be taught.

Lots of similar applications:

- E.g. assigning wavelengths to cell phone conversations to avoid interference.
  - Vertices are conversations
  - Edges between "nearby" conversations
  - Colors are wavelengths.
- Scheduling final exams
  - Vertices are courses
  - Edges between courses with overlapping enrollment
  - Colors are exam times.

Chromatic Number

The *chromatic number* of a graph G, written $\chi(G)$, is the smallest number of colors needed to color it so that no two adjacent vertices have the same color.

A graph G is *k-colorable* if $k \geq \chi(G)$.

Determining $\chi(G)$

Some observations:

- If G is a complete graph with n vertices, $\chi(G) = n$
  - A complete graph is one where there is an edge between every pair of nodes.
  - How many edges are there in a complete graph with n nodes?
- If G has a clique of size k, then $\chi(G) \geq k$.
  - A *clique* of size k in a graph G is a completely connected subgraph of G with k vertices.
  - Let $c(G)$ be the *clique number* of G: the size of the largest clique in G. Then
$$\chi(G) \geq c(G).$$
- If $\Delta(G)$ is the maximum degree of any vertex, then
$$\chi(G) \leq \Delta(G) + 1 :$$
  - Color G one vertex at a time; color each vertex with the "smallest" color not used for a colored vertex adjacent to it.

How hard is it to determine if $\chi(G) \leq k$?

- It's NP complete, just like
  - determining if $c(G) \geq k$
  - determining if G has a Hamiltonian path
  - determining if a propositional formula is satisfiable

Can guess and verify.

The Four-Color Theorem

Can a map be colored with four colors, so that no countries with common borders have the same color?

- This is an instance of graph coloring
  - The vertices are countries
  - Two vertices are joined by an edge if the countries they represent have a common border

A *planar graph* is one where all the edges can be drawn on a plane (piece of paper) without any edges crossing.

- The graph of a map is planar

Four-Color Theorem: Every map can be colored using at most four colors so that no two countries with a common boundary have the same color.

- Equivalently: every planar graph is four-colorable

Four-Color Theorem: History

- First conjectured by Galton (Darwin's cousin) in 1852
- False proofs given in 1879, 1880; disproved in 1891
- Computer proof given by Appel and Haken in 1976
  - They reduced it to 1936 cases, which they checked by computer
- Proof simplified in 1996 by Robertson, Sanders, Seymour, and Thomas
  - But even their proof requires computer checking
  - They also gave an $O(n^2)$ algorithm for four coloring a planar graph
- Proof checked by Coq theorem prover (Werner and Gonthier) in 2004
  - So you don't have to trust the proof, just the theorem prover

Note that the theorem doesn't apply to countries with non-contiguous regions (like U.S. and Alaska).

Bipartite Graphs

A graph $G(V, E)$ is *bipartite* if we can partition V into disjoint sets V_1 and V_2 such that all the edges in E joins a vertex in V_1 to one in V_2.

- A graph is bipartite iff it is 2-colorable
- Everything in V_1 gets one color, everything in V_2 gets the other color.

Example: Suppose we want to represent the "is or has been married to" relation on people. Can partition the set V of people into males (V_1) and females (V_2). Edges join two people who are or have been married.

Example: We can represent the "has taken a course from" relation by taking the nodes to be professors and students with an edge between s and t if student s has taken a course from professor t. Is this bipartite?

Number of Sexual Partners

University of Chicago study (1994);

- Men have on average 74% more opposite-gender partners than women

ABC News poll (2004):

- average man has 20 (opposite=gender) partners in his lifetime, the average woman has 6
  - Margin of error 2.5%!

New York Times (2007)

- Men had seven partners while women had 4

Who is right? Are these numbers even possible?

Characterizing Bipartite Graphs

Theorem: G is bipartite iff G has no odd-length cycles.

Proof: Suppose that G is bipartite, and it has edges only between V_1 and V_2. Suppose, to get a contradiction, that $(x_0, x_1, \ldots, x_{2k}, x_0)$ is an odd-length cycle. If $x_0 \in V_1$, then x_2 is in V_1. An easy induction argument shows that $x_{2i} \in V_1$ and $x_{2i+1} \in V_2$ for $0 = 1, \ldots, k$. But then the edge between x_{2k} and x_0 is an edge between two nodes in V_1; contradiction!

- Get a similar contradiction if $x_0 \in V_2$.

Conversely, suppose $G(V, E)$ has no odd-length cycles.

- Partition the vertices in V into two sets V_1 and V_2 as follows:
  - Start at an arbitrary vertex x_0; put it in V_1.
  - Put all the vertices one step from x_0 into V_2
  - Put all the vertices two steps from x_0 into V_1;
  - ...

This construction works if all nodes are reachable from x.

- What if some node y isn't reachable from x?

Characterizing Bipartite Graphs

Theorem: G is bipartite iff G has no odd-length cycles.

Proof: Suppose that G is bipartite, and it has edges only between V_1 and V_2. Suppose, to get a contradiction, that $(x_0, x_1, \ldots, x_{2k}, x_0)$ is an odd-length cycle. If $x_0 \in V_1$, then x_2 is in V_1. An easy induction argument shows that $x_{2i} \in V_1$ and $x_{2i+1} \in V_2$ for $0 = 1, \ldots, k$. But then the edge between x_{2k} and x_0 is an edge between two nodes in V_1; contradiction!

- Get a similar contradiction if $x_0 \in V_2$.

Conversely, suppose $G(V, E)$ has no odd-length cycles.

- Partition the vertices in V into two sets V_1 and V_2 as follows:
  - Start at an arbitrary vertex x_0; put it in V_1.
  - Put all the vertices one step from x_0 into V_2
  - Put all the vertices two steps from x_0 into V_1;
  - ...

This construction works if all nodes are reachable from x.

- What if some node y isn't reachable from x?
  - Repeat the process, starting at y

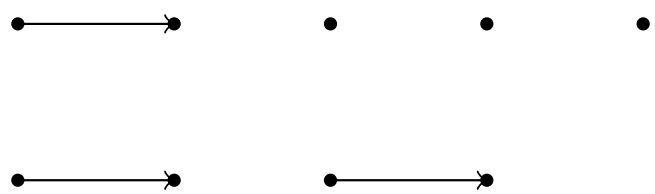
This gives a polynomial-time algorithm to check if G is bipartite.

Matchings

A *matching* in a graph $G = (V, E)$ is a subset M of E no vertex in v is on more than one edge in M

- $M = \emptyset$ is a (not so interesting) matching

Here's another matching:



M is a *perfect matching* if every vertex in V is on an edge in M.

- Marriage defines a matching
  - At least, if you don't allow polygamy or polyandry

  But it's not a perfect matching!

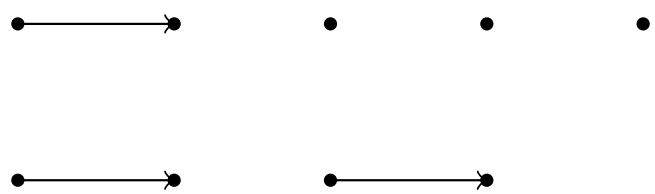
We are particularly interested in finding maximal matches in bipartite graphs (as many people as possible are married).

Matchings

A *matching* in a graph $G = (V, E)$ is a subset M of E no vertex in v is on more than one edge in M

- $M = \emptyset$ is a (not so interesting) matching

Here's another matching:



M is a *perfect matching* if every vertex in V is on an edge in M.

- Marriage defines a matching
  - At least, if you don't allow polygamy or polyandry

  But it's not a perfect matching!

We are particularly interested in finding maximal matches in bipartite graphs (as many people as possible are married).

Given a graph $G = (V, E)$ and $W \subseteq V$, define

$$N(W) = \{t : (s, t) \in E, s \in W\}.$$

$N(W)$ = the neighbors of nodes in W

Hall's Theorem

Theorem: (Hall's Theorem) If $G = (V, E)$ is a bipartite graph with edges from V_1 to V_2, there is a matching M that *covers* V_1 (i.e., every vertex in V_1 is incident to an edge in M) iff, for every subset $W \subseteq V_1$, $|W| \leq |N(W)|$. It's a perfect matching if $|V_1| = |V_2|$.

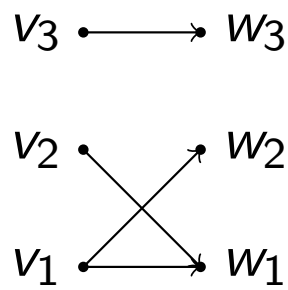
Hall's Theorem

Theorem: (Hall's Theorem) If $G = (V, E)$ is a bipartite graph with edges from V_1 to V_2, there is a matching M that *covers* V_1 (i.e., every vertex in V_1 is incident to an edge in M) iff, for every subset $W \subseteq V_1$, $|W| \leq |N(W)|$. It's a perfect matching if $|V_1| = |V_2|$.

Proof: If there is a matching V that covers all the vertices in V_1 then clearly $|V'| \leq |N(V')|$ for every subset $V' \subseteq V_1$.
For the converse, we proceed by strong induction on $|V_1|$. The base case ($|V_1| = 1$) is trivial.
For the inductive step, there are two cases:

- $|W| < |N(W)|$ for all $W \subseteq V_1$: Choose $v_1 \in V_1$ and $v_2 \in V_2$ such that $\{v_1, v_2\} \in E$. Add $\{v_1, v_2\}$ to the matching M and repeat the process for $G' = (V', E')$, where $V' = V - \{v_1, v_2\}$, $V_1' = V_1 - \{v_1\}$, $V_2' = V_2 - \{v_2\}$, and E' is the result of removing all edges involving v_1 or v_2 from E.

- If $|W| = |N(W)|$ for some $W \subseteq V_1$, this won't work (the induction hypothesis isn't maintained: may have $|W| > |N(W)|$). Example:



Note that $N(\{v_1, v_2\}) = \{w_1, w_2\}$, so $|N(\{v_1, v_2\})| = |\{v_1, v_2\}|$.

  - If we remove v_1 and w_1, the induction hypothesis no longer holds
  - v_2 will not be matched with anything.

Solution: By the induction hypotheis, there is a matching between W and $N(W)$.

  - Can remove $W \cup N(W)$ and all edges leading to and from W and $N(W)$.
  - This maintains the induction hypothesis!