Graph Theory

MAT230

Discrete Mathematics

Fall 2019

Outline

Definitions

Definition

A **graph** $G = (V, E)$ consists of a set V of **vertices** (also called **nodes**) and a set E of **edges**.

Definition

If an edge connects to a vertex we say the edge is **incident** to the vertex and say the vertex is an **endpoint** of the edge.

Definition

If an edge has only one endpoint then it is called a **loop edge**.

Definition

If two or more edges have the same endpoints then they are called **multiple** or **parallel** edges.

Definitions

Definition

Two vertices that are joined by an edge are called **adjacent** vertices.

Definition

A **pendant** vertex is a vertex that is connected to exactly one other vertex by a single edge.

Definition

A **walk** in a graph is a sequence of alternating vertices and edges $v_1 e_1 v_2 e_2 \ldots v_n e_n v_{n+1}$ with $n \geq 0$. If $v_1 = v_{n+1}$ then the walk is **closed**. The **length** of the walk is the number of edges in the walk. A walk of length zero is a **trivial walk**.

Definitions

Definition

A **trail** is a walk with no repeated edges. A **path** is a walk with no repeated vertices. A **circuit** is a closed trail and a **trivial circuit** has a single vertex and no edges. A trail or circuit is **Eulerian** if it uses every edge in the graph.

Definition

A **cycle** is a nontrivial circuit in which the only repeated vertex is the first/last one.

Definition

A **simple graph** is a graph with no loop edges or multiple edges. Edges in a simple graph may be specified by a set $\{v_i, v_j\}$ of the two vertices that the edge makes adjacent. A graph with more than one edge between a pair of vertices is called a **multigraph** while a graph with loop edges is called a **pseudograph**.

Definitions

Definition

A **directed graph** is a graph in which the edges may only be traversed in one direction. Edges in a simple directed graph may be specified by an ordered pair (v_i, v_j) of the two vertices that the edge connects. We say that v_i is **adjacent to** v_j and v_j is **adjacent from** v_i.

Definition

The **degree** of a vertex is the number of edges incident to the vertex and is denoted $\deg(v)$.

Definition

In a directed graph, the **in-degree** of a vertex is the number of edges **incident to** the vertex and the **out-degree** of a vertex is the number of edges **incident from** the vertex.

Definitions

Definition

A graph is **connected** if there is a walk between every pair of distinct vertices in the graph.

Definition

A graph H is a **subgraph** of a graph G if all vertices and edges in H are also in G.

Definition

A **connected component** of G is a connected subgraph H of G such that no other connected subgraph of G contains H.

Definition

A graph is called **Eulerian** if it contains an Eulerian circuit.

Definitions

Definition

A **tree** is a connected, simple graph that has no cycles. Vertices of degree 1 in a tree are called the **leaves** of the tree.

Definition

Let G be a simple, connected graph. The subgraph T is a **spanning tree of** G if T is a tree and every node in G is a node in T.

Definition

A **weighted graph** is a graph $G = (V, E)$ along with a function $w : E \to \mathbb{R}$ that associates a numerical weight to each edge. If G is a weighted graph, then T is a **minimal spanning tree of** G if it is a spanning tree and no other spanning tree of G has smaller total weight.

Definitions

Definition

The **complete graph** on n nodes, denoted K_n, is the simple graph with nodes $\{1, \ldots, n\}$ and an edge between every pair of distinct nodes.

Definition

A graph is called **bipartite** if its set of nodes can be partitioned into two disjoint sets S_1 and S_2 so that every edge in the graph has one endpoint in S_1 and one endpoint in S_2.

Definition

The **complete bipartite graph** on n, m nodes, denoted $K_{n,m}$, is the simple bipartite graph with nodes $S_1 = \{a_1, \ldots, a_n\}$ and $S_2 = \{b_1, \ldots, b_m\}$ and with edges connecting each node in S_1 to every node in S_2.

Definitions

Definition

Simple graphs G and H are called **isomorphic** if there is a bijection f from the nodes of G to the nodes of H such that $\{v, w\}$ is an edge in G if and only if $\{f(v), f(w)\}$ is an edge of H. The function f is called an **isomorphism**.

Definition

A simple, connected graph is called **planar** if there is a way to draw it on a plane so that no edges cross. Such a drawing is called an **embedding** of the graph in the plane.

Definition

For a planar graph G embedded in the plane, a **face** of the graph is a region of the plane created by the drawing. The area of the plane outside the graph is also a face, called the unbounded face.

Theorems

Theorem

Let G be a connected graph. Then G is Eulerian if and only if every vertex in G has even degree.

Theorem (Handshaking Lemma)

In any graph with n vertices v_i and m edges

$$\sum_{i=1}^{n} \deg(v_i) = 2m$$

Corollary

A connected non-Eulerian graph has an Eulerian trail if and only if it has exactly two vertices of odd degree. The trail begins and ends these two vertices.

Theorems

Theorem

If T is a tree with n edges, then T has $n+1$ vertices.

Theorem

Two graphs that are isomorphic to one another must have

1. *The same number of nodes.*
2. *The same number of edges.*
3. *The same number of nodes of any given degree.*
4. *The same number of cycles.*
5. *The same number of cycles of any given size.*

Theorems

Theorem (Kuratowski's Theorem)

A graph G is nonplanar if and only if it contains a "copy" of $K_{3,3}$ or K_5 as a subgraph.

Theorem (Euler's Formula for Planar Graphs)

For any connected planar graph G embedded in the plane with V vertices, E edges, and F faces, it must be the case that

$$V + F = E + 2.$$

Graphs vs Plots

Recall that a graph consists of two sets: a set of vertices and a set of edges.

While we often represent graphs visually, we can distinguish between a graph and a **plot** in the following way: A graph stores information and connections between information while a plot provides a visual representation of the information stored in a graph.

Given that graphs are important, we now examine how we can represent graphs using a computer and see how one computer package handles graphs.

A Quick Matrix Review

A **matrix** is a rectangular array of numbers. A matrix with m **rows** and n **columns** said to be an $m \times n$ matrix.

Entries in the matrix are addressed by their row and column numbers. Given a matrix A, the entry a_{ij} is in the i^{th} row and j^{th} column of A. Notice that we always list the row index first.

We say a matrix A is **symmetric** if $a_{ji} = a_{ij}$.

Not Symmetric

$$\begin{bmatrix} 0 & 5 & 2 & 1 \\ 1 & 3 & 0 & 1 \\ 4 & 6 & 8 & 3 \\ 0 & 7 & 3 & 1 \end{bmatrix}$$

Symmetric

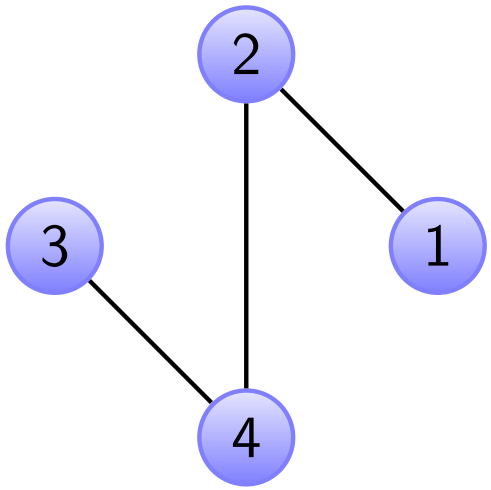
$$\begin{bmatrix} 0 & 3 & 7 & 1 \\ 3 & 8 & 5 & 0 \\ 7 & 5 & 2 & 4 \\ 1 & 0 & 4 & 0 \end{bmatrix}$$

Adjacency Matrices

Let G be a graph with n vertices. We can use an $n \times n$ matrix to store the graph. Let

$$g_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is adjacent to vertex } j \\ 0 & \text{if vertex } i \text{ is not adjacent to vertex } j \end{cases}$$

For example, the graph on the left has the adjacency matrix on the right.



$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Note: matrix is symmetric

The adjacency matrix for a directed graph will not be symmetric unless the directed graph itself is symmetric.

Sparse Graphs and Matrices

Consider K_{30}, the complete graph with 30 vertices. This graph has $C(30, 2) = 435$ edges since every vertex is connected to every other vertex. The adjacency matrix will have 1's in every non-diagonal position (why not on the diagonals?). We say the 30×30 adjacency matrix is **dense** or **full** since most of the entries are non-zero.

Now consider C_{30}, a **cycle** (or **ring**) graph with 30 vertices. Each vertex is connected to two other vertices to form a single ring or cycle. This means there are only 30 edges. So, while the adjacency matrix will be 30×30, only 60 entries in it will be non-zero. In this case we say the graph and the adjacency matrix are **sparse**.

Adjacency Matrix Examples

Adjacency matrix for K_8

8×8 matrix with 64 elements

$2 \cdot C(8,2) = 56$ non-zero entries

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Adjacency matrix for C_8

8×8 matrix with 64 elements

8 non-zero entries

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Edge Lists

Note that in the case of undirected graphs we really only need the upper-right triangle or the lower-left triangle (about half) of the adjacency matrix to store the information in the graph.

In the case of directed graphs, we need the full matrix.

In either case, if the graph is sparse, there are more efficient ways to store the graph. One of these is to maintain an **edge list**.

Edge Lists

1. **Edge List v1**: For each vertex, store a list of vertices that the vertex is adjacent to.
2. **Edge List v2**: Store all edges in the graph as a list of ordered pairs.

Adjacency Matrix

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Edge List v1

$$\{\{2\}, \{1,4\}, \{4\}, \{2,3\}\}$$

Edge List v2

$$\{(1,2), (2,1), (2,4), (3,4), (4,2), (4,3)\}$$

Which version we choose to use is largely dependent on how we need to access the information.

A walk around Gordon's campus

Is it possible to start in front of the Ken Olsen Science Center (labeled E), walk along every pathway on Gordon's main campus exactly once, and end up back in front of the Science Center?

A walk around Gordon's campus

- What do you think the answer is?

A walk around Gordon's campus

- What do you think the answer is?
- How could you find out?

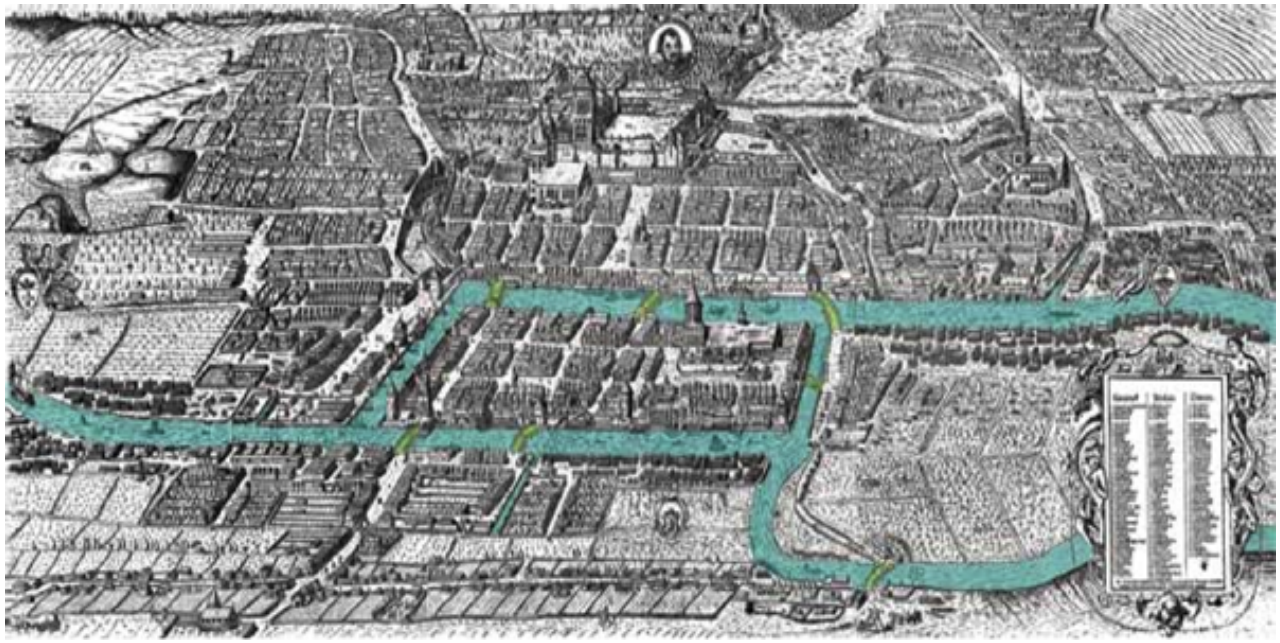
A walk around Gordon's campus

- What do you think the answer is?
- How could you find out?
- Have you ever wondered this before?

A walk around Gordon's campus

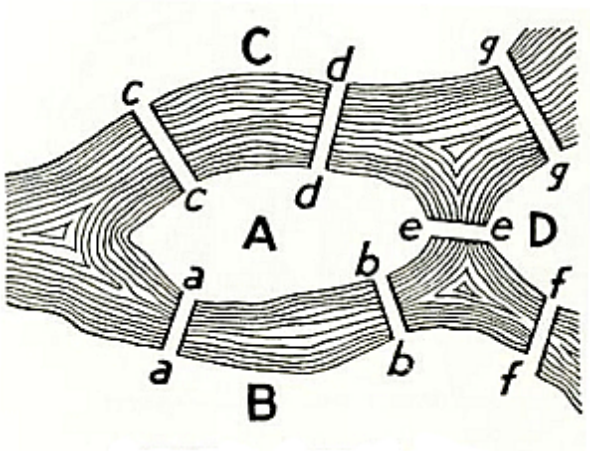
- What do you think the answer is?
- How could you find out?
- Have you ever wondered this before?
- This same question has been considered before (although not specifically about Gordon's campus)...

The Königsberg bridge problem



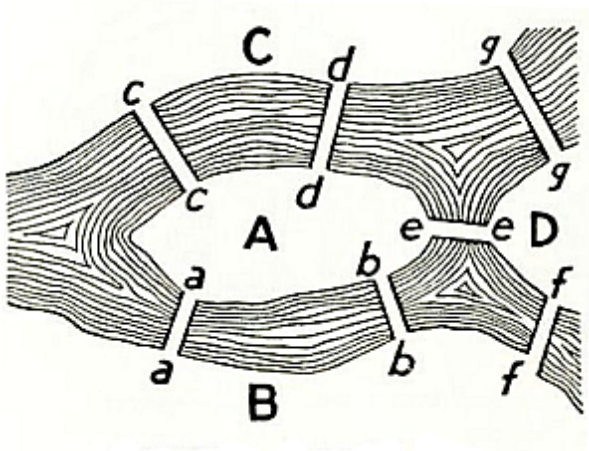
The town of Königsberg, Prussia (now a city in Russia called Kaliningrad) is built on the both banks of the river Preger as well as on an island in the river. At one time there were seven bridges linking one bank to the other as well as both banks to the island. The people in the town wondered if were possible to start at some point in the town, walk about the town crossing each bridge exactly once, and end up back at the starting point.

The Königsberg bridge problem



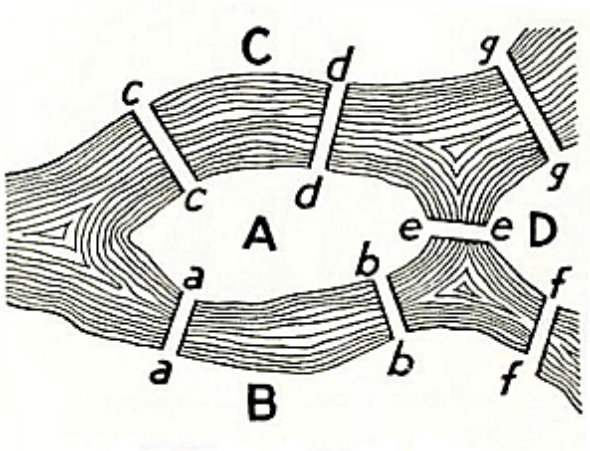
- A, B, C, and D label regions of land (A and D are islands).

The Königsberg bridge problem



- A, B, C, and D label regions of land (A and D are islands).
- Bridges are labeled with a, b, c, etc.

The Königsberg bridge problem



- A, B, C, and D label regions of land (A and D are islands).
- Bridges are labeled with a, b, c, etc.
- Try to find a route, starting anywhere you want, that crosses every bridge once and end up back where you started.

Don't worry

- Chances are you were not able to find a solution.

Don't worry

- Chances are you were not able to find a solution.
- Don't worry; you're in good company.

Don't worry

- Chances are you were not able to find a solution.
- Don't worry; you're in good company.
- Leonhard Euler, the brilliant Swiss mathematician, was also unable to find such a route.

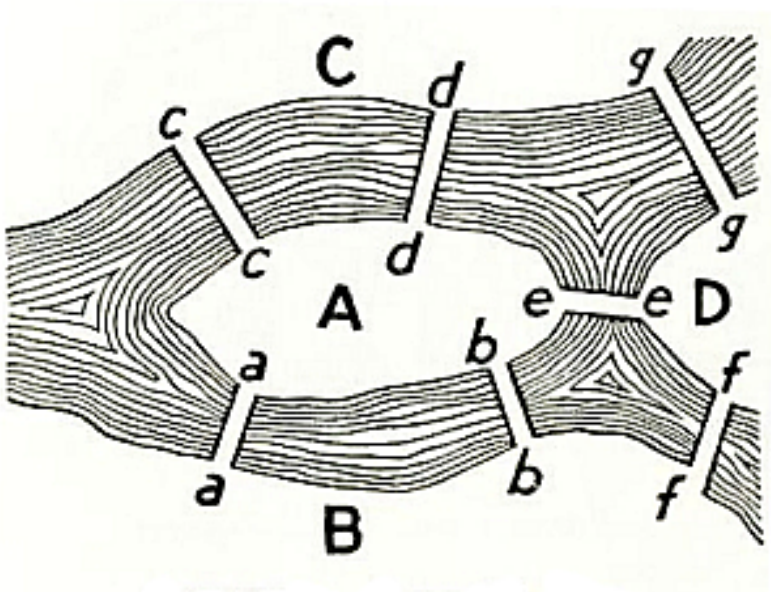
Don't worry

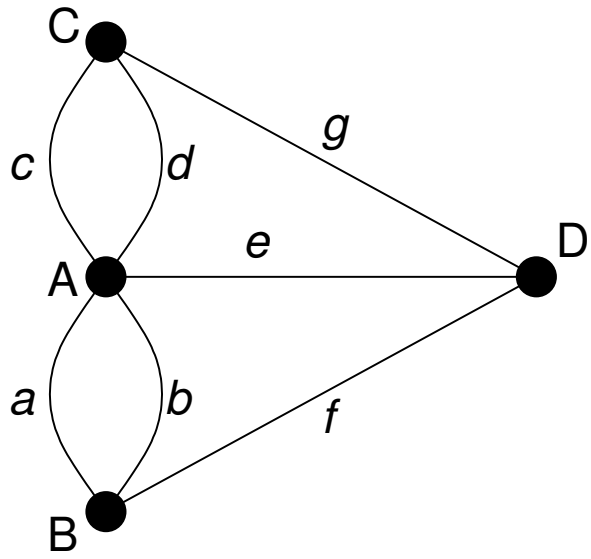
- Chances are you were not able to find a solution.
- Don't worry; you're in good company.
- Leonhard Euler, the brilliant Swiss mathematician, was also unable to find such a route.

- Not content with this result, Euler figured out how to show for certain that no such route existed.

Königsberg bridge graph

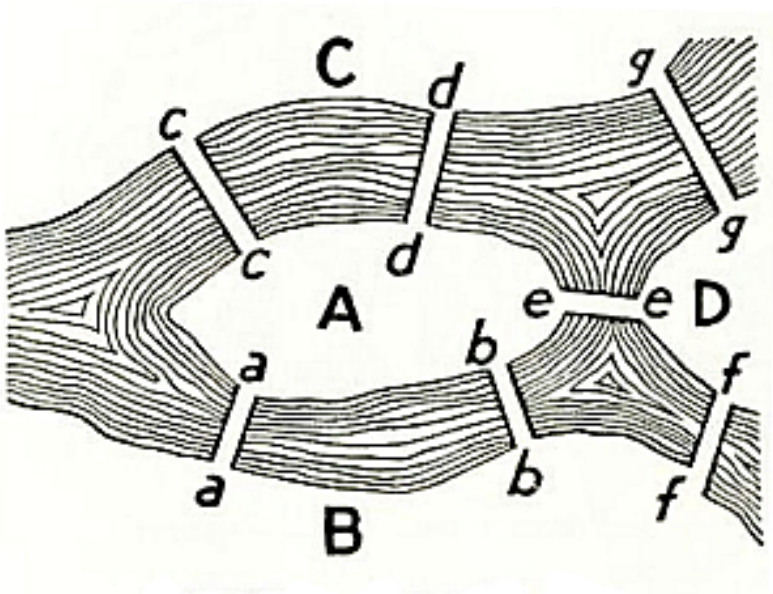
- If we focus on the essential parts of this problem, we can construct a graph.

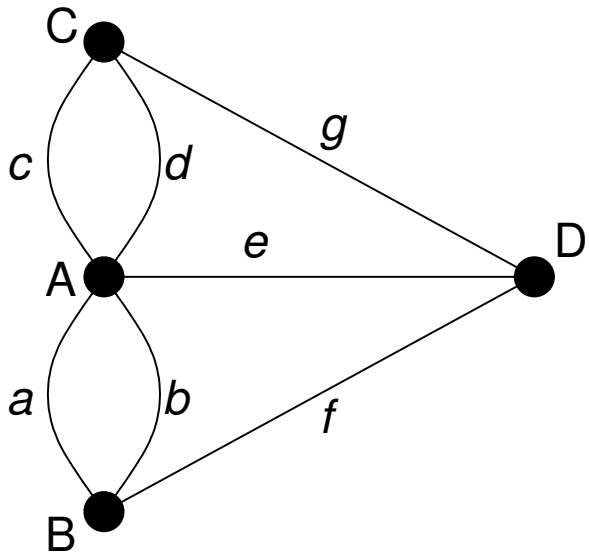




Königsberg bridge graph

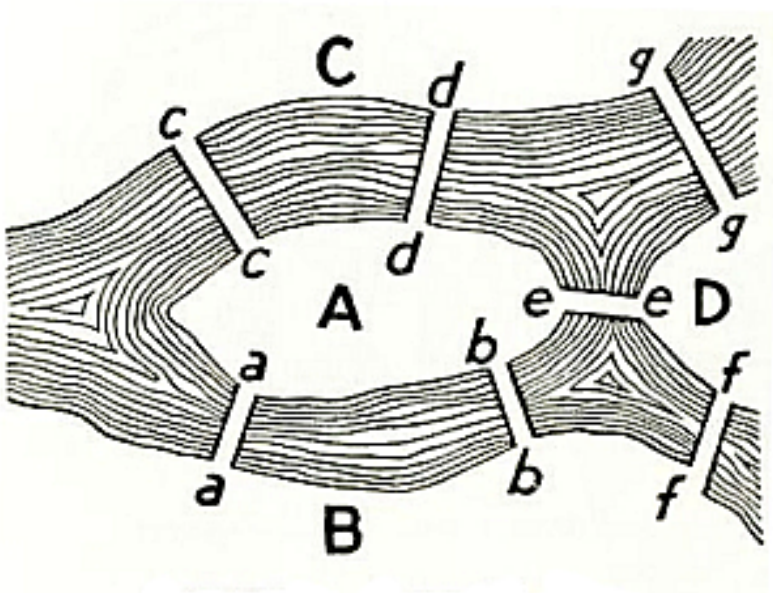
- If we focus on the essential parts of this problem, we can construct a graph.
- This *abstraction* is simpler than the bridge picture and yet contains all the necessary information.

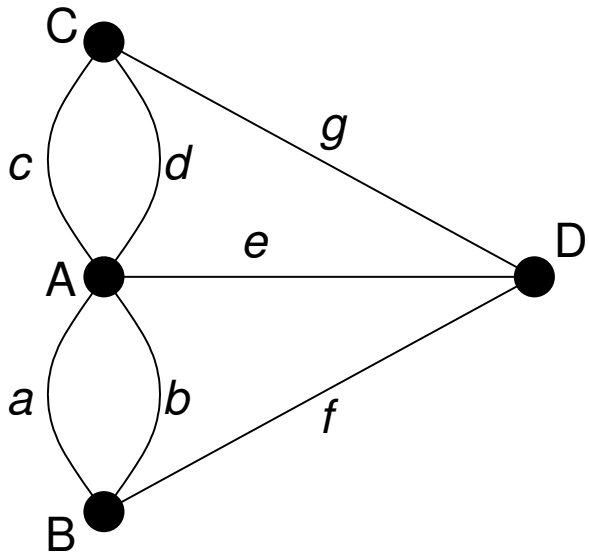




Königsberg bridge graph

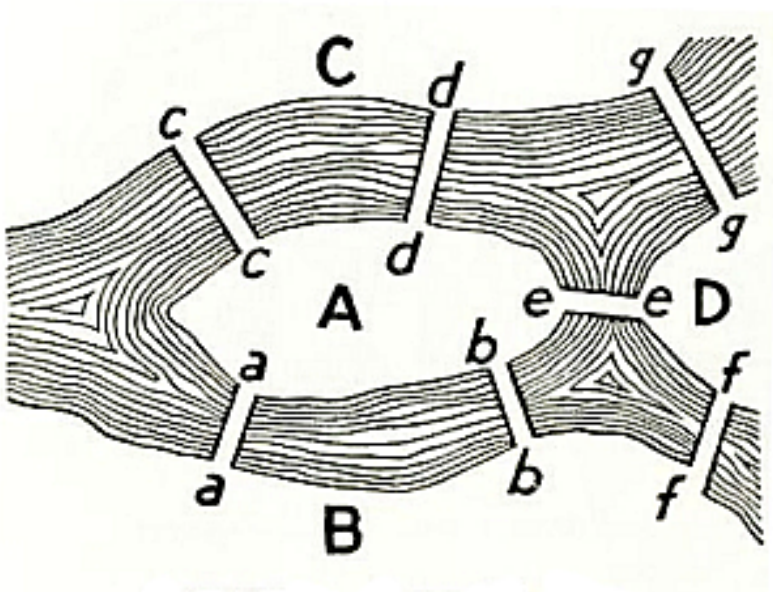
- If we focus on the essential parts of this problem, we can construct a graph.
- This *abstraction* is simpler than the bridge picture and yet contains all the necessary information.
- Each region of land is represented by a vertex.

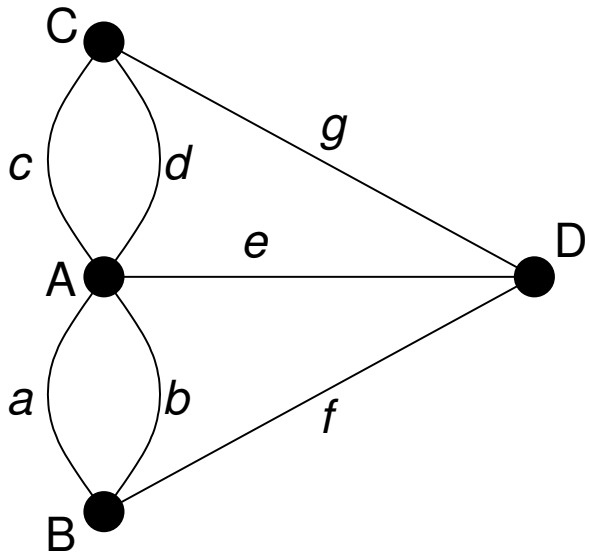




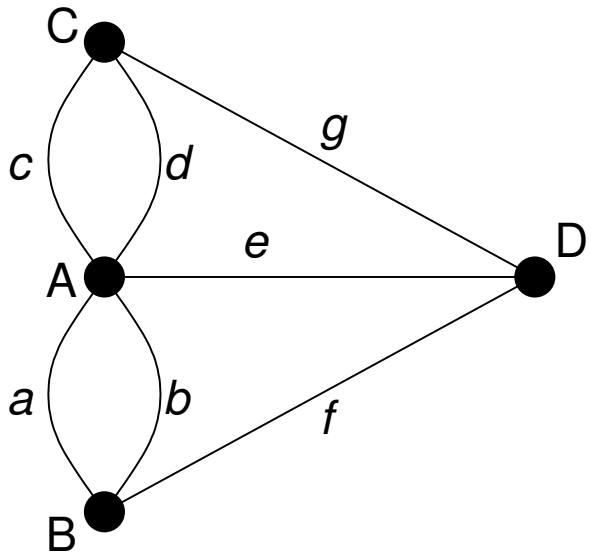
Königsberg bridge graph

- If we focus on the essential parts of this problem, we can construct a graph.
- This *abstraction* is simpler than the bridge picture and yet contains all the necessary information.
- Each region of land is represented by a vertex.
- Each bridge connecting two regions corresponds to an edge.



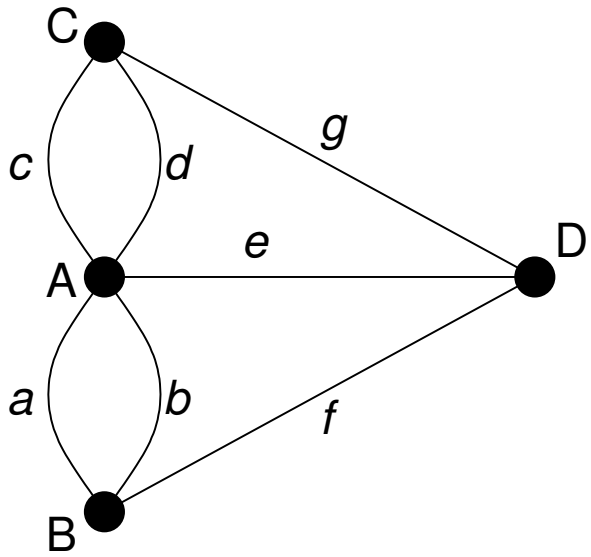


Königsberg bridge graph



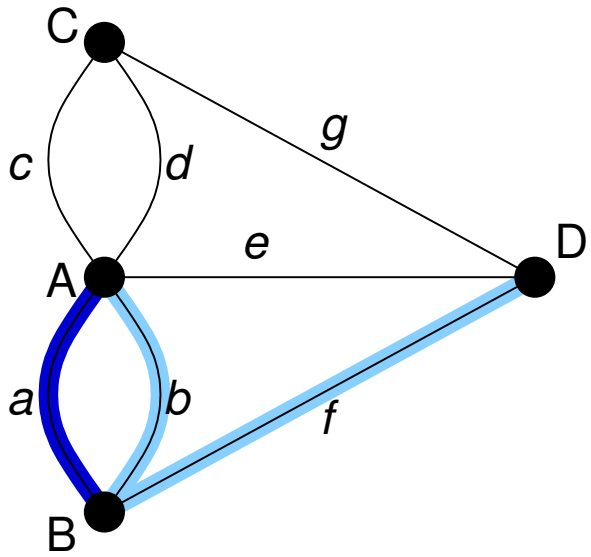
- Consider starting at A then visiting B, D, and C, before returning to A. This does not use each edge, but does visit each vertex.

Königsberg bridge graph



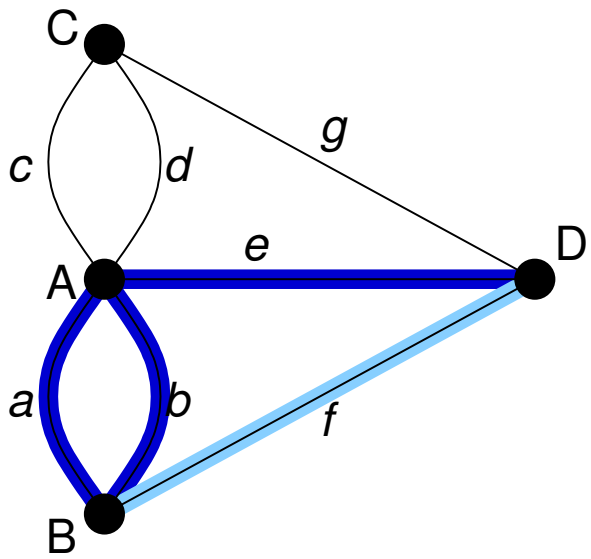
- Consider starting at A then visiting B, D, and C, before returning to A. This does not use each edge, but does visit each vertex.
- We could show this with the walk AaBfDgCcA.

Königsberg bridge graph



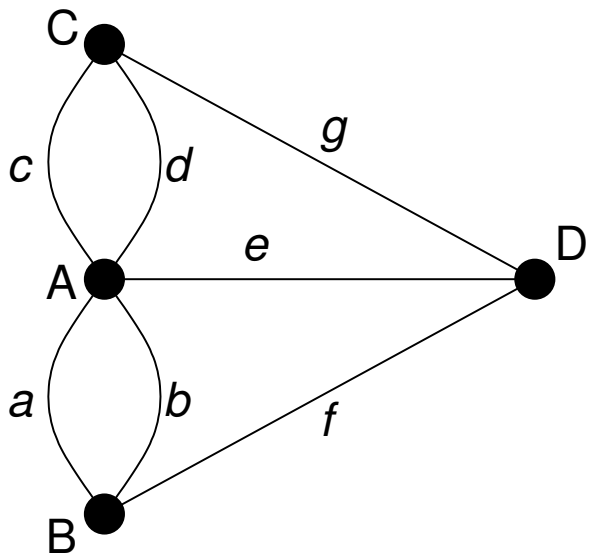
- Suppose we now wanted to use every edge in the graph. Starting at A we can move to B. We can now either return to A or visit D.

Königsberg bridge graph



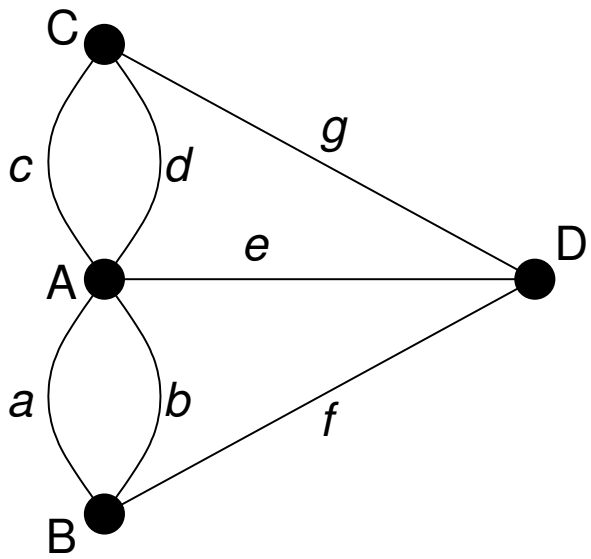
- Suppose we now wanted to use every edge in the graph. Starting at A we can move to B. We can now either return to A or visit D.
- Notice, however, that we will eventually return to B along the third edge connecting to it (since we're trying to use all edges) but then there is no fourth edge to exit from B.

Königsberg bridge graph



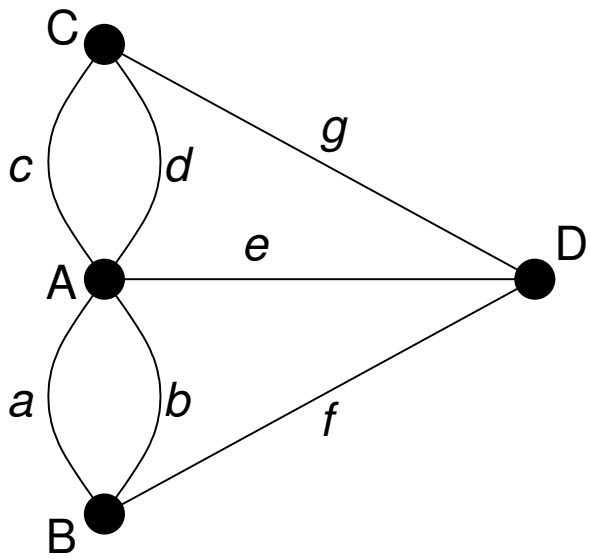
- To have any hope of traversing each edge attached to a vertex there must be an even number of edges attached to the vertex; these form an *entry-exit* pair.

Königsberg bridge graph



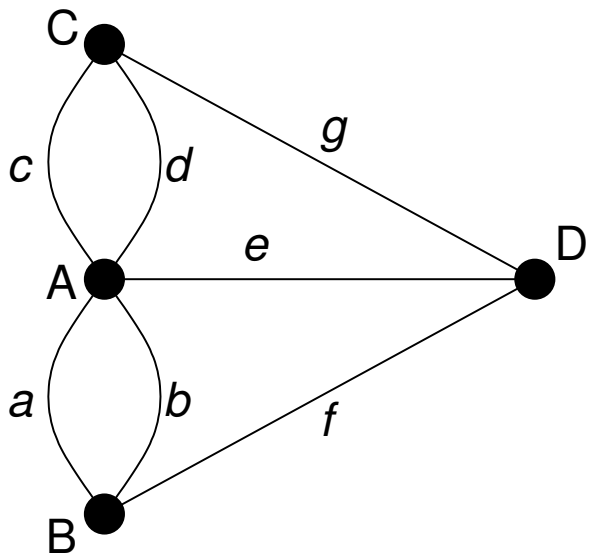
- To have any hope of traversing each edge attached to a vertex there must be an even number of edges attached to the vertex; these form an *entry-exit* pair.
- This is also true for the starting vertex, except one pair of edges is an *exit-entry* pair.

Königsberg bridge graph



- Notice what we've established: if there is a vertex with an odd number of edges attached to it, we will be prevented from finding a route that uses all edges once and returns to the starting point.

Königsberg bridge graph



- Notice what we've established: if there is a vertex with an odd number of edges attached to it, we will be prevented from finding a route that uses all edges once and returns to the starting point.
- If every vertex has an even number of edges attached to it, then there is always an *entry-exit* pair so we can find a route.

Review of Definitions

Before continuing, it is helpful to review some definitions:

Review of Definitions

Before continuing, it is helpful to review some definitions:

- A **walk** is a sequence of alternating vertices and edges. A **trail** is a walk with no repeated edges and a **path** is a walk with no repeated vertices.

Review of Definitions

Before continuing, it is helpful to review some definitions:

- A **walk** is a sequence of alternating vertices and edges. A **trail** is a walk with no repeated edges and a **path** is a walk with no repeated vertices.
- A **circuit** is a closed trail; a trail that starts and stops at the same vertex.

Review of Definitions

Before continuing, it is helpful to review some definitions:

- A **walk** is a sequence of alternating vertices and edges. A **trail** is a walk with no repeated edges and a **path** is a walk with no repeated vertices.
- A **circuit** is a closed trail; a trail that starts and stops at the same vertex.
- An **Eulerian circuit** in a graph is a circuit that contains every edge of the graph.

Review of Definitions

Before continuing, it is helpful to review some definitions:

- A **walk** is a sequence of alternating vertices and edges. A **trail** is a walk with no repeated edges and a **path** is a walk with no repeated vertices.
- A **circuit** is a closed trail; a trail that starts and stops at the same vertex.
- An **Eulerian circuit** in a graph is a circuit that contains every edge of the graph.
- An **Eulerian trail** in a graph is a trail that contains every edge of the graph.

Review of Definitions

Before continuing, it is helpful to review some definitions:

- A **walk** is a sequence of alternating vertices and edges. A **trail** is a walk with no repeated edges and a **path** is a walk with no repeated vertices.
- A **circuit** is a closed trail; a trail that starts and stops at the same vertex.
- An **Eulerian circuit** in a graph is a circuit that contains every edge of the graph.
- An **Eulerian trail** in a graph is a trail that contains every edge of the graph.
- A graph is called an **Eulerian Graph** if it has an Eulerian circuit.

Euler's Theorem

Theorem (Euler's Theorem)

Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

Euler's Theorem

Theorem (Euler's Theorem)

Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

1. Let C be an empty circuit in G, assign $G_0 = G$, and set $k = 0$.

Euler's Theorem

Theorem (Euler's Theorem)

Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

1. Let C be an empty circuit in G, assign $G_0 = G$, and set $k = 0$.
2. Find any circuit C_k in G_k.

Euler's Theorem

Theorem (Euler's Theorem)

Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

1. Let C be an empty circuit in G, assign $G_0 = G$, and set $k = 0$.
2. Find any circuit C_k in G_k.
3. Merge circuit C_k into circuit C.

Euler's Theorem

Theorem (Euler's Theorem)

Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

1. Let C be an empty circuit in G, assign $G_0 = G$, and set $k = 0$.
2. Find any circuit C_k in G_k.
3. Merge circuit C_k into circuit C.
4. Construct graph G_{k+1} from G_k by removing edges in circuit C_k.

Euler's Theorem

Theorem (Euler's Theorem)

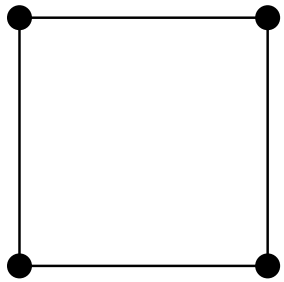
Let G be an connected, undirected graph. G has an Eulerian circuit if and only if every vertex in G has even degree. G has an Eulerian trail if and only if G has exactly two vertices with odd degree.

Basic Algorithm to find an Eulerian Circuit in G:

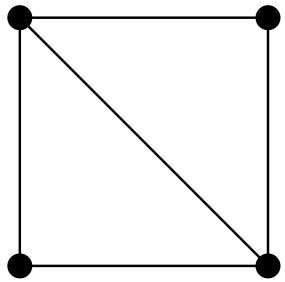
1. Let C be an empty circuit in G, assign $G_0 = G$, and set $k = 0$.
2. Find any circuit C_k in G_k.
3. Merge circuit C_k into circuit C.
4. Construct graph G_{k+1} from G_k by removing edges in circuit C_k.
5. If C does not contain all edges in G then increment k and go to Step 2.

Finding Eulerian Circuits and Trails

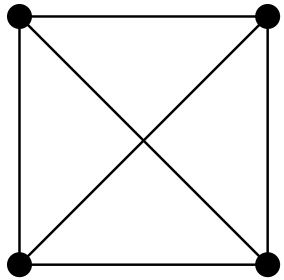
Which of the following graphs have an Eulerian circuit? If you can't find a Eulerian circuit, can you find an Eulerian trail?



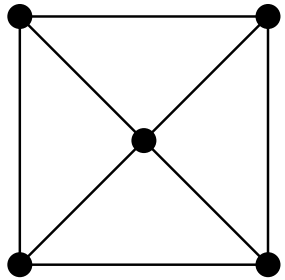
A



B



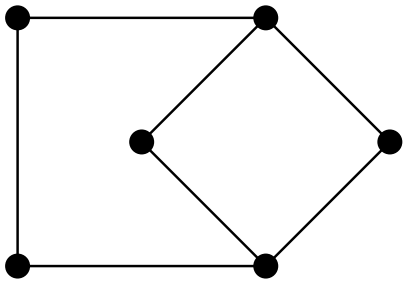
C



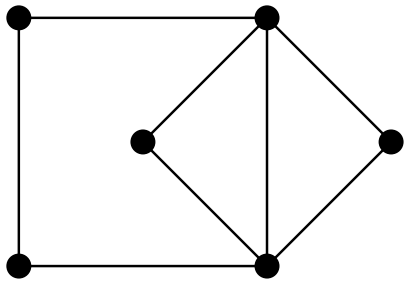
D

Finding Eulerian Circuits and Trails

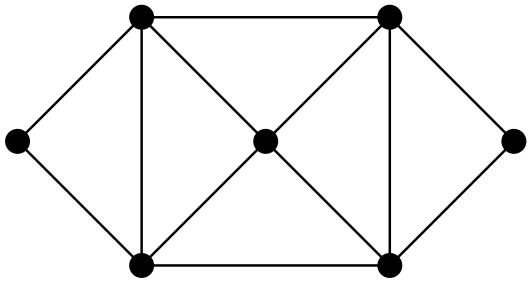
How about these? Any Eulerian circuits or trails?



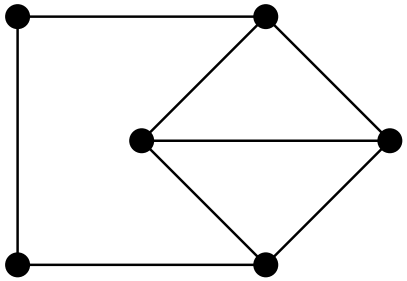
E



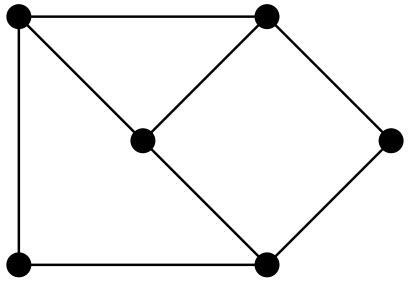
F



I



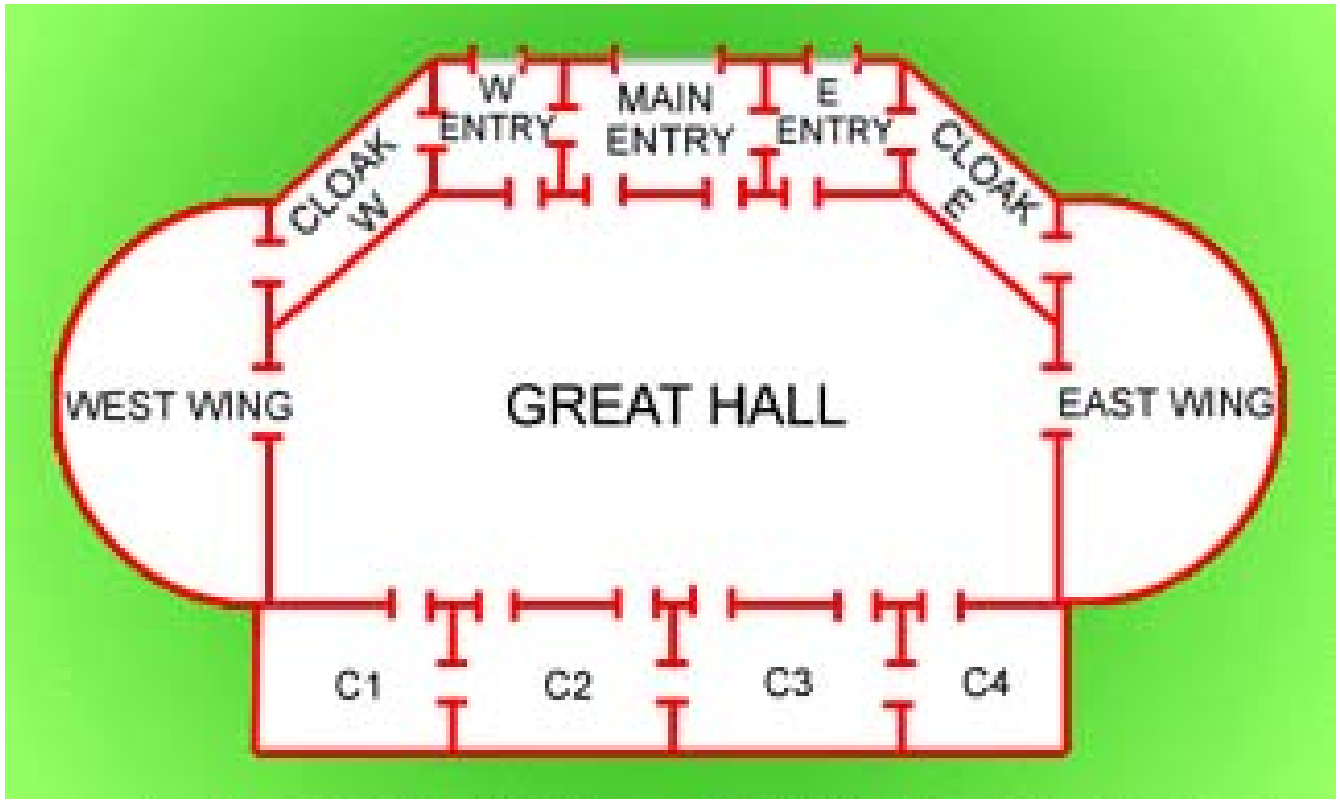
G



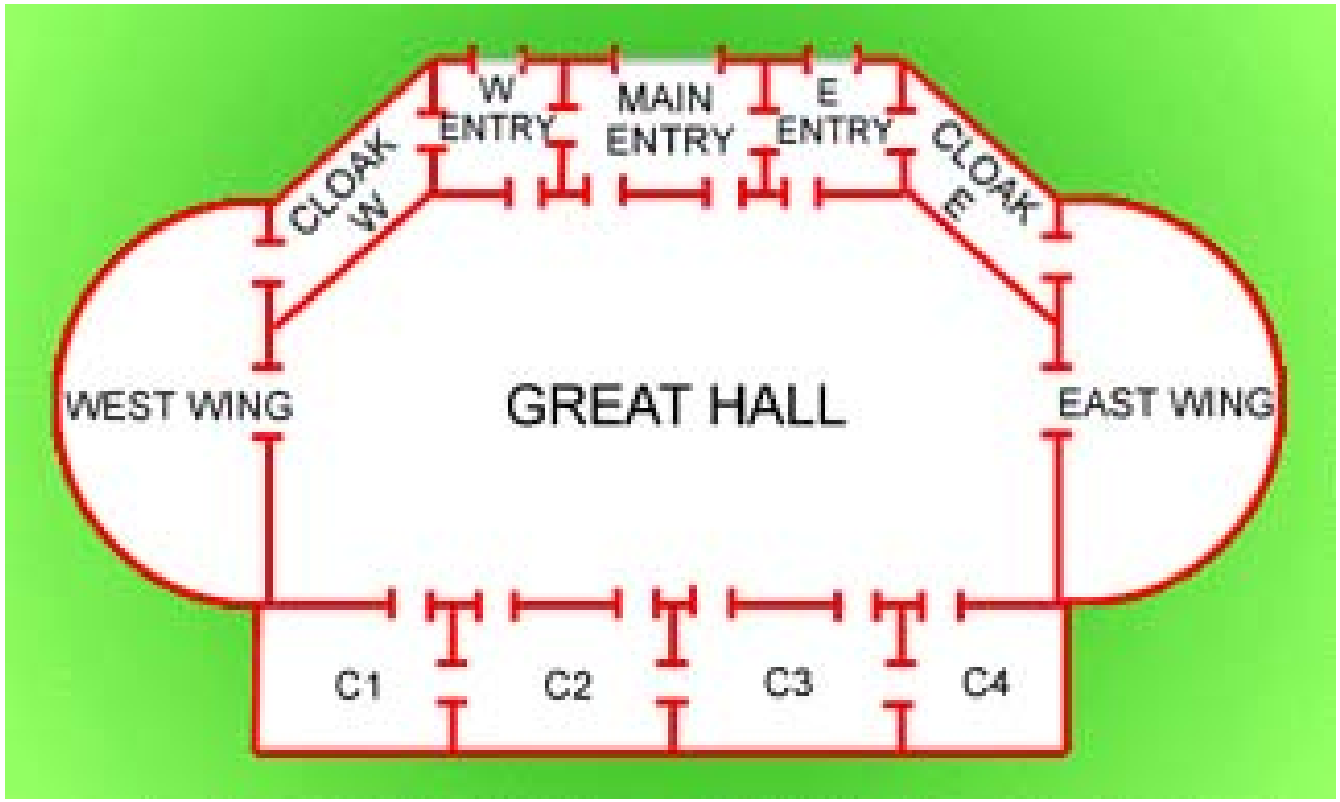
H

Finding Eulerian Circuits and Trails

Consider the floor plan shown here. Is it possible to walk through and around this building passing through each and every doorway exactly once?

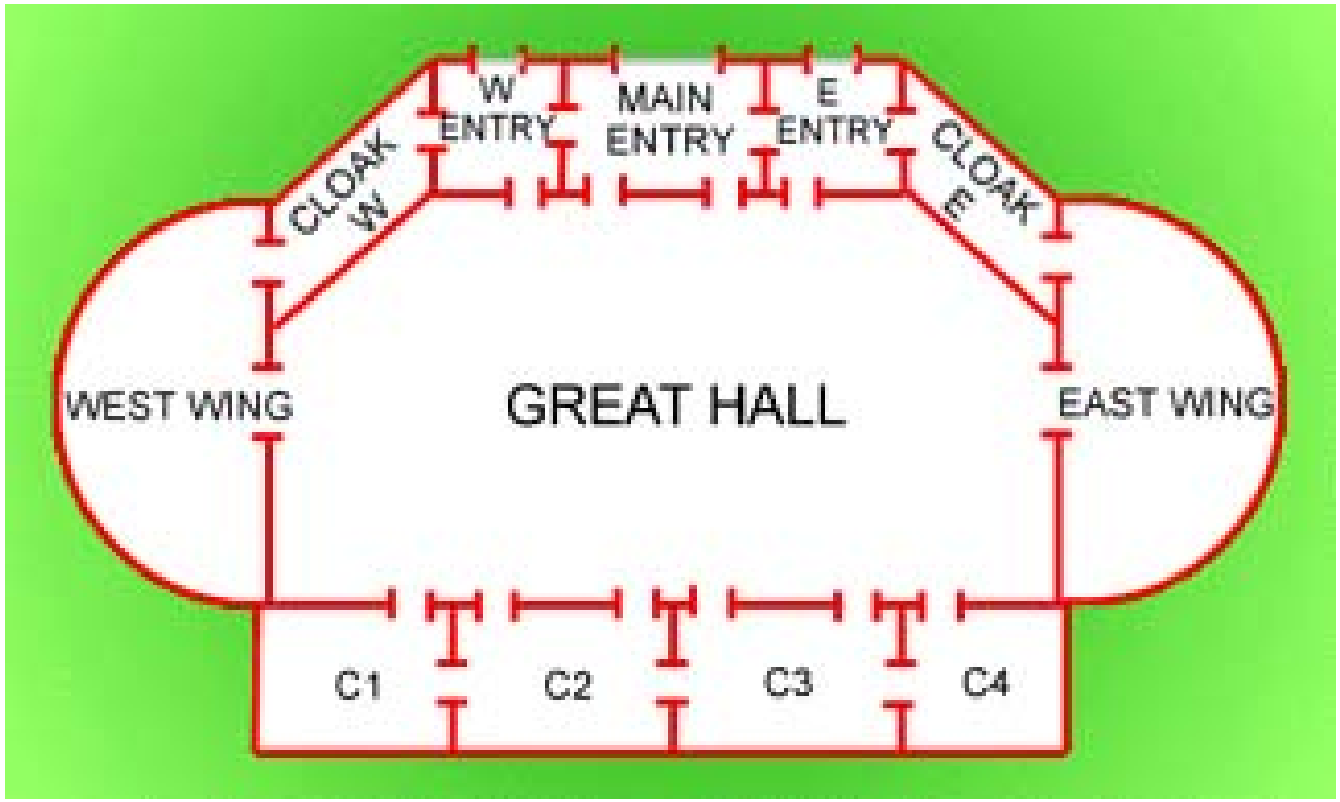


Finding Eulerian Circuits and Trails



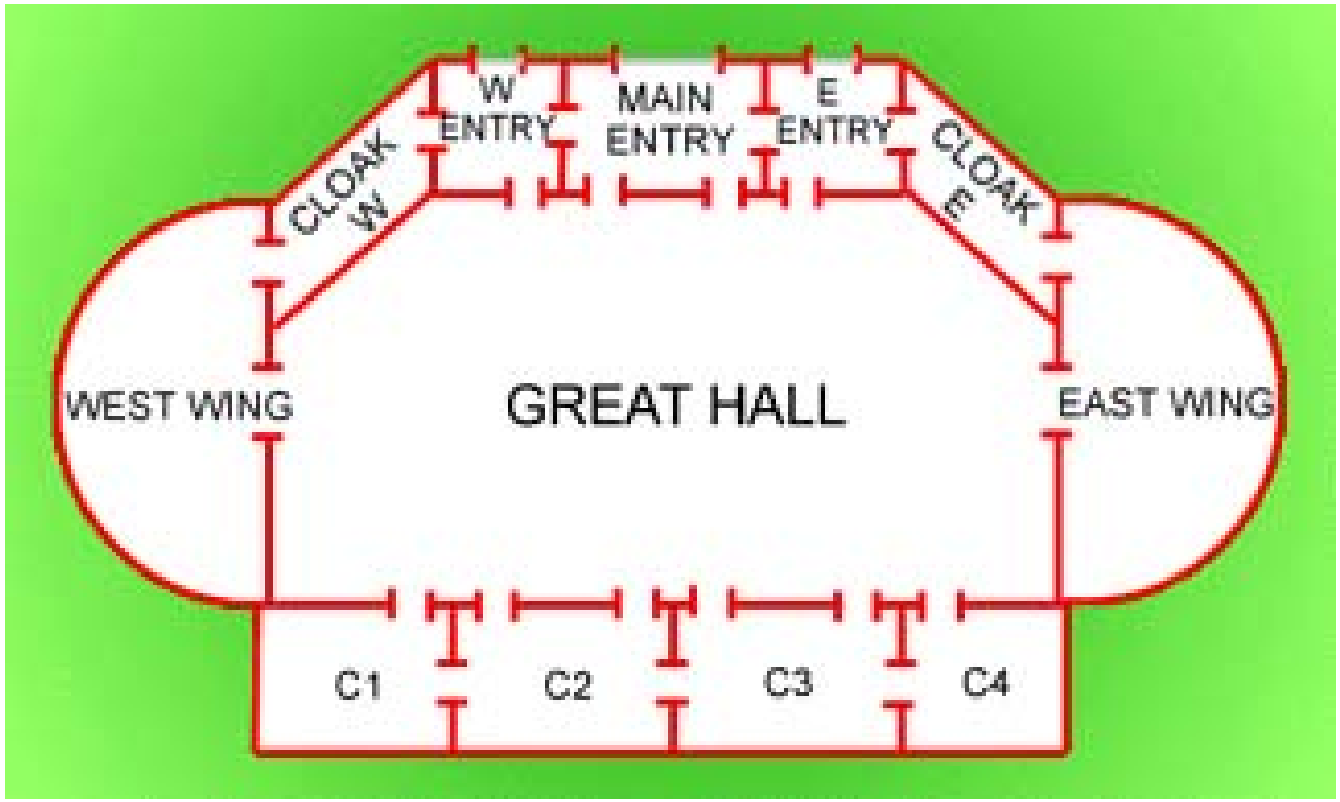
1. Start by creating any circuit.

Finding Eulerian Circuits and Trails



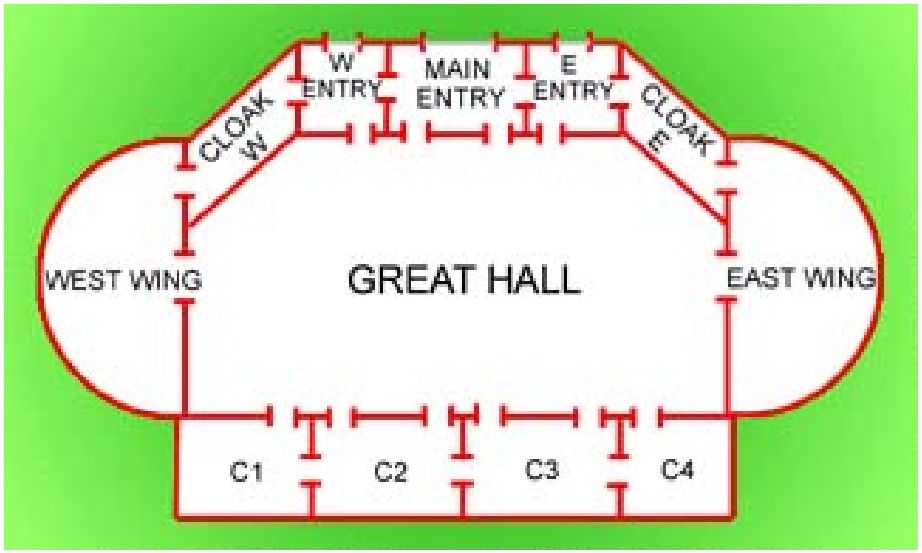
1. Start by creating any circuit.
2. Pick any vertex with unused edges and find a circuit using it. Merge this new circuit into the previously found circuit.

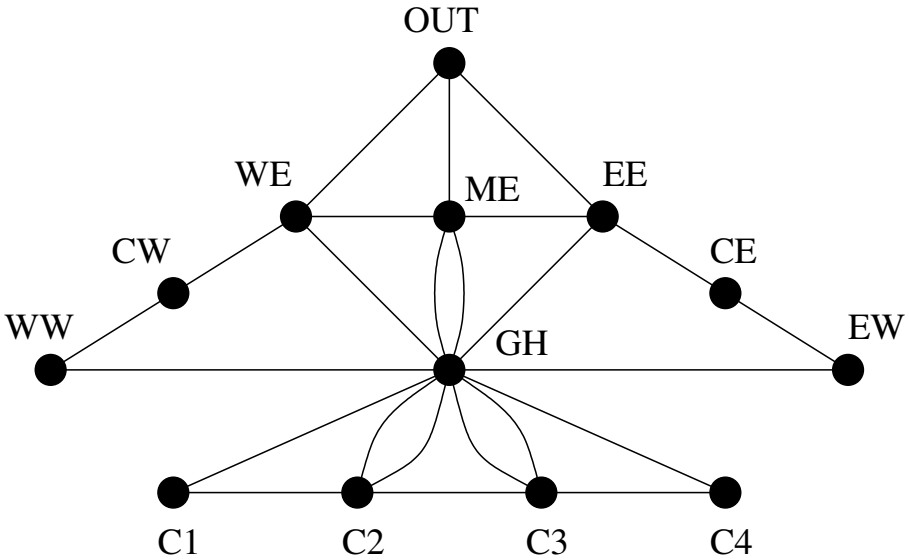
Finding Eulerian Circuits and Trails



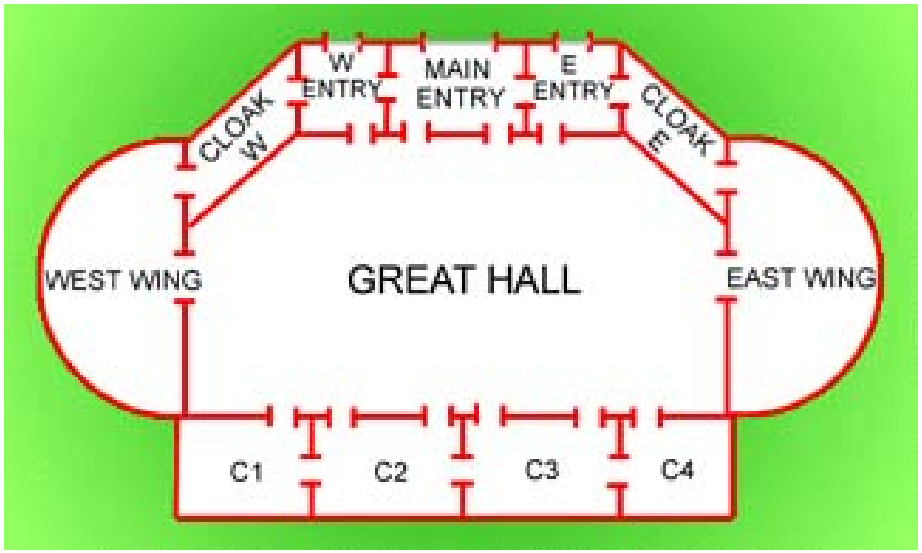
1. Start by creating any circuit.
2. Pick any vertex with unused edges and find a circuit using it. Merge this new circuit into the previously found circuit.
3. Continue the above steps until an Eulerian circuit or trail is found.

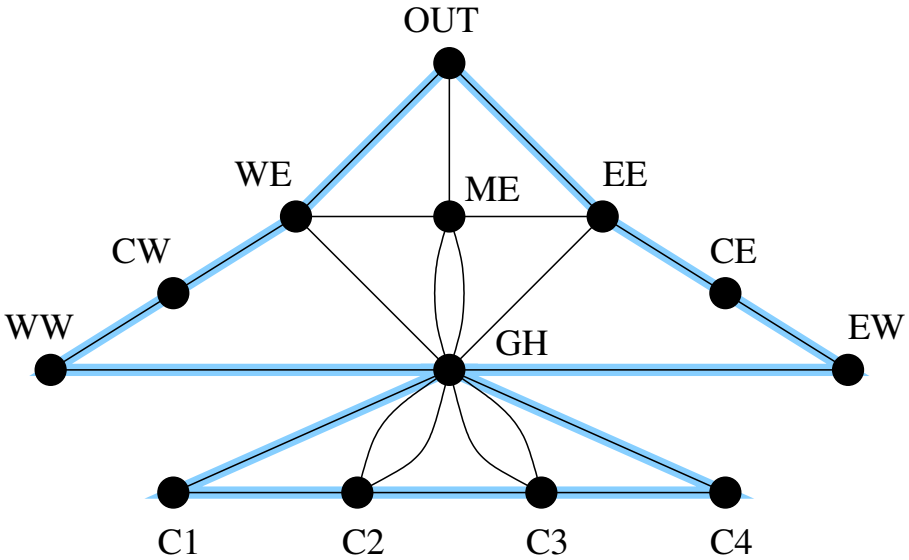
Finding Eulerian Circuits and Trails



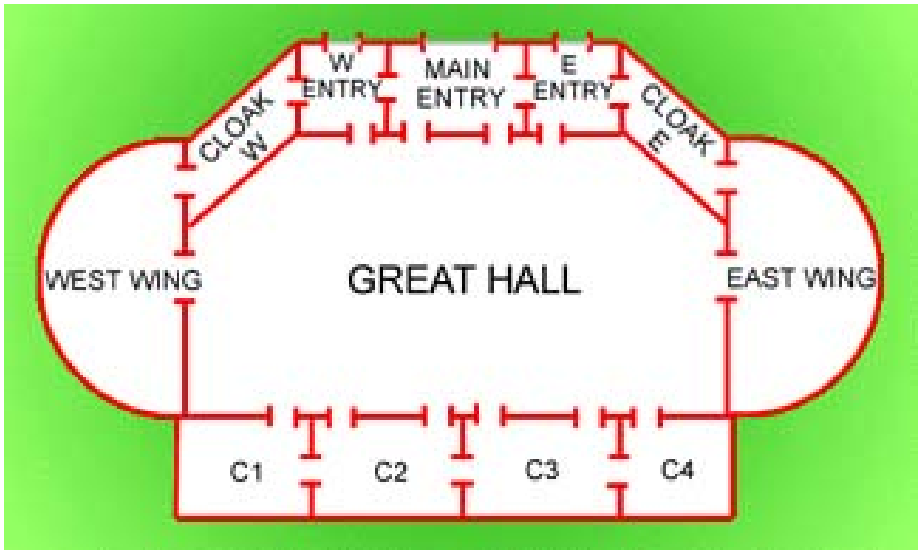


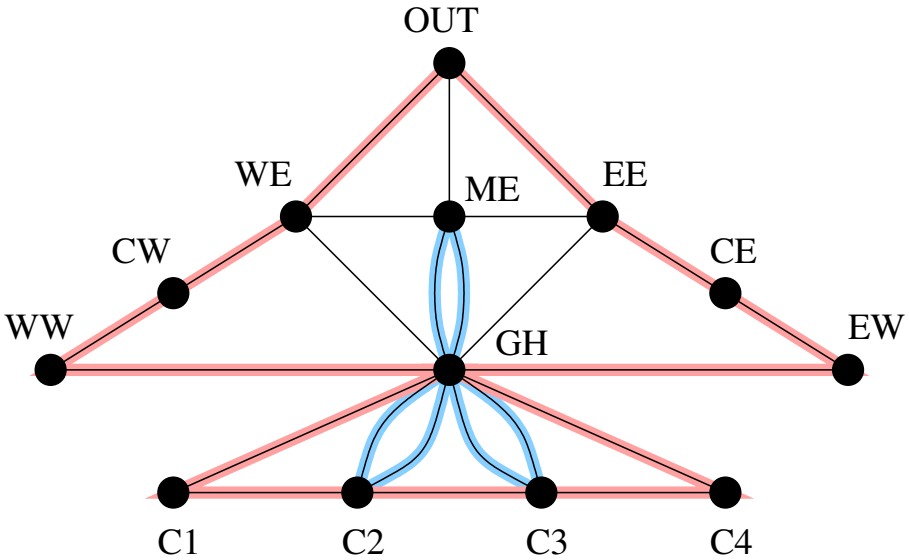
Finding Eulerian Circuits and Trails



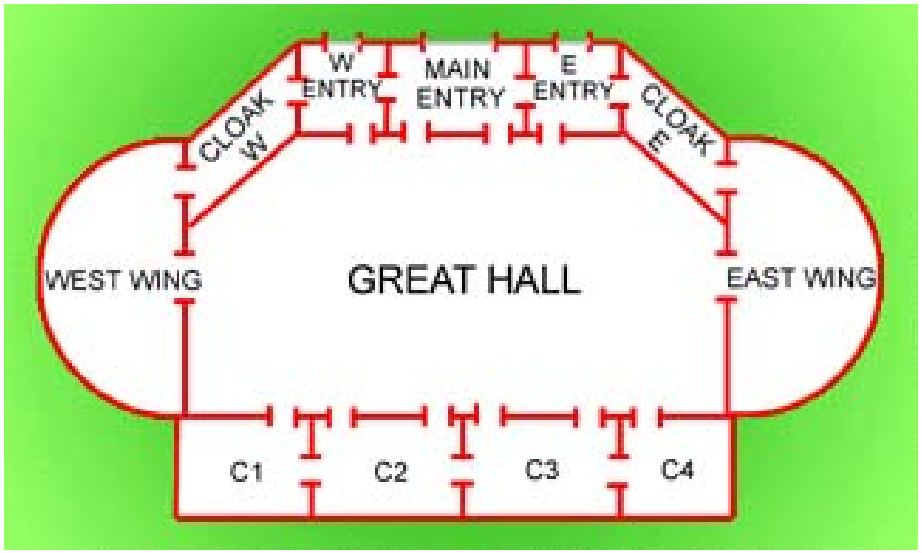


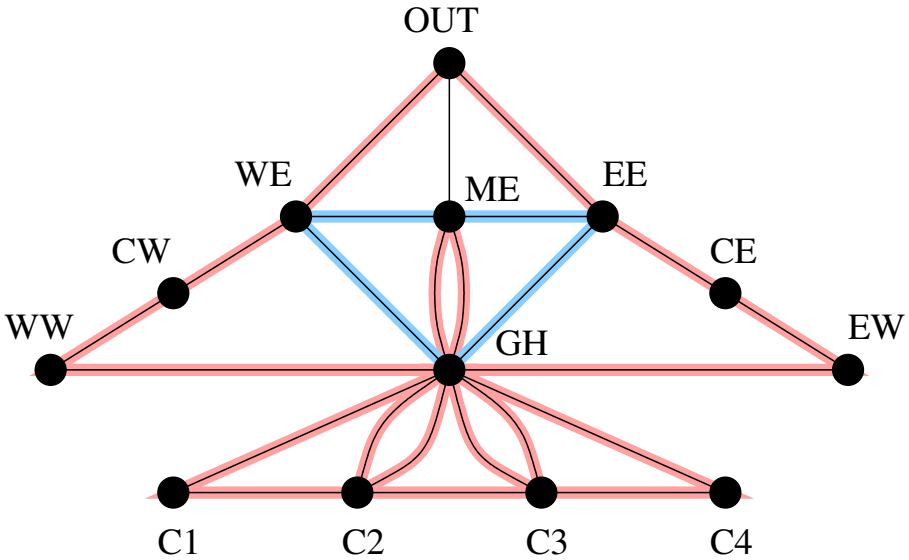
Finding Eulerian Circuits and Trails



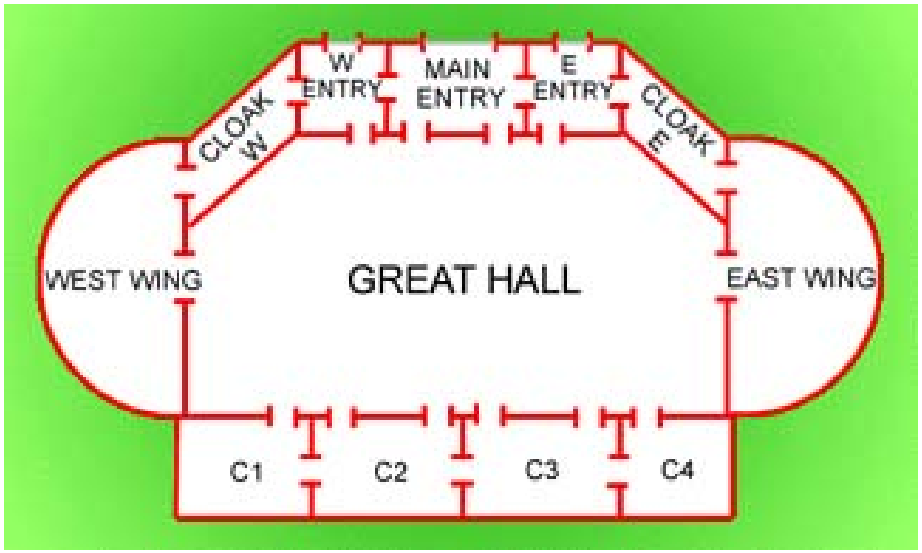


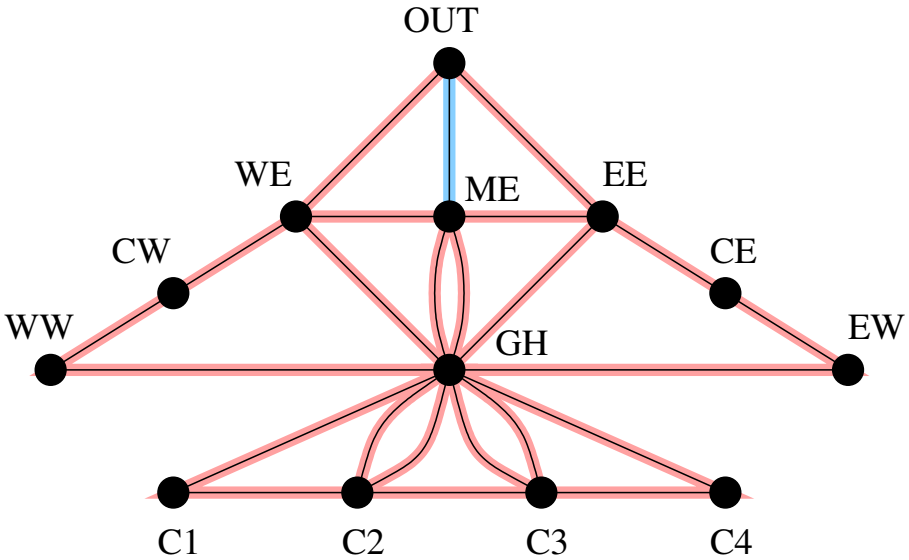
Finding Eulerian Circuits and Trails





Finding Eulerian Circuits and Trails





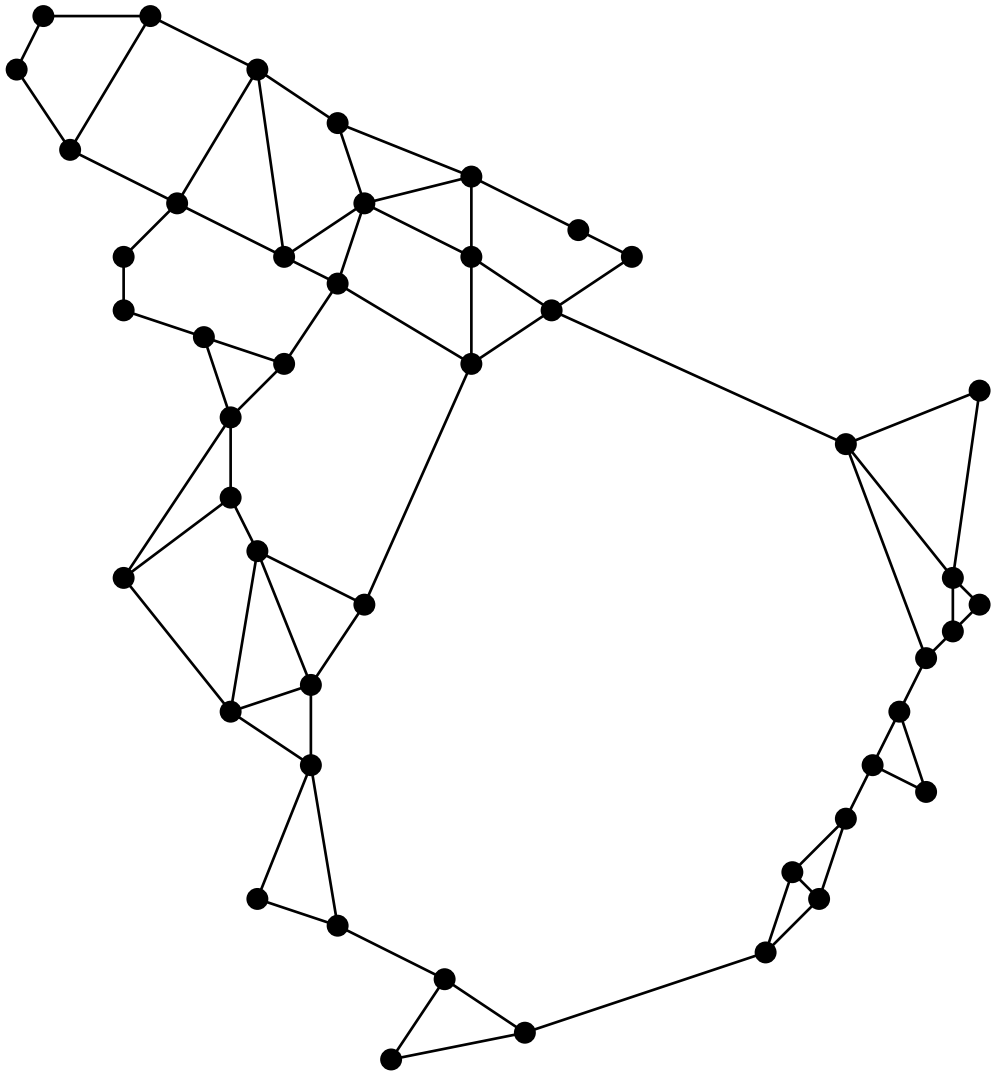
Both OUT and ME have an odd number of incident edges; we can start at one and end at the other. Thus, we have found an Eulerian trail.

A Walk Around Gordon's Campus

Finally, back to our original question: Is it possible to start in front of the Ken Olsen Science Center (labeled E), walk along every pathway on Gordon's main campus exactly once, and end up back in front of the Science Center?

A Walk Around Gordon's Campus

A somewhat simplified graph of most of the walkways on campus may help:



Hamiltonian Circuits and Trails

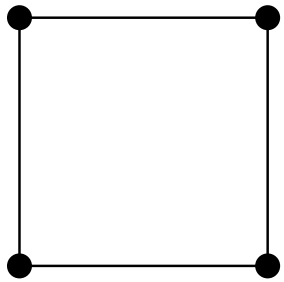
There is another interesting question we can ask about graph: Given a connected graph G, does G have a circuit in which every vertex is visited exactly once?

Such a circuit, if it exists, is called a **Hamiltonian circuit**. Hamiltonian trails are defined similarly.

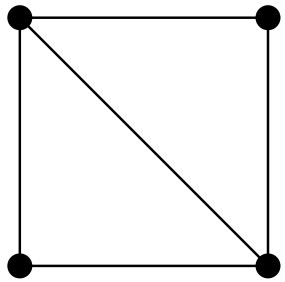
It turns out that finding a Hamiltonian circuit or trail in a graph is quite difficult in general, but can be done for many simple graphs.

Finding Hamiltonian Circuits and Trails

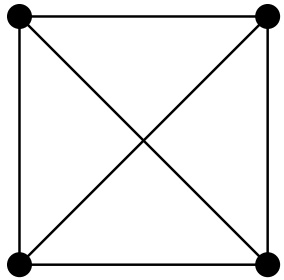
Which of the following graphs have an Hamiltonian circuit? If you can't find a Hamiltonian circuit, can you find an Hamiltonian trail?



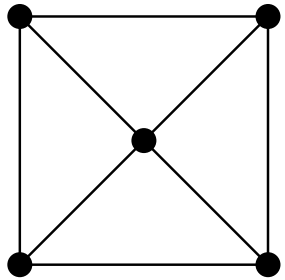
A



B



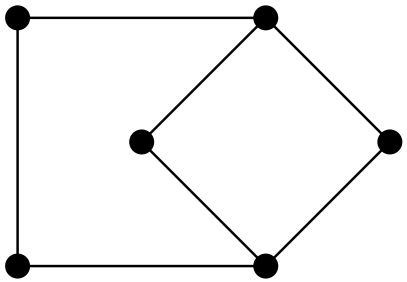
C



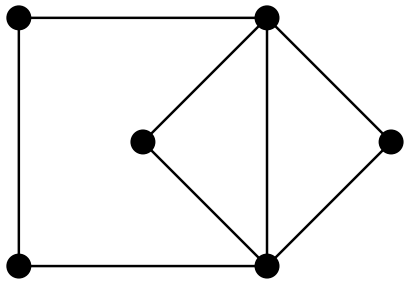
D

Finding Hamiltonian Circuits and Trails

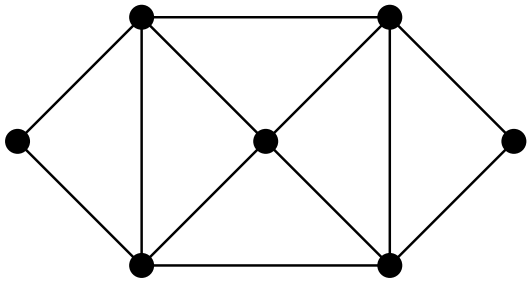
How about these? Any Hamiltonian circuits or trails?



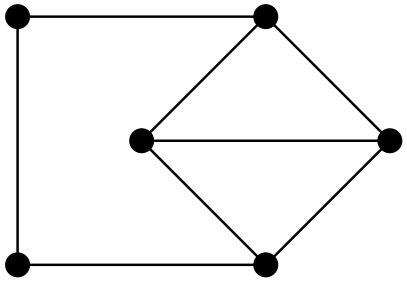
E



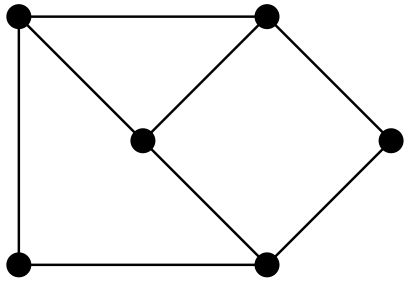
F



I



G



H

Finding Hamiltonian Circuits and Trails

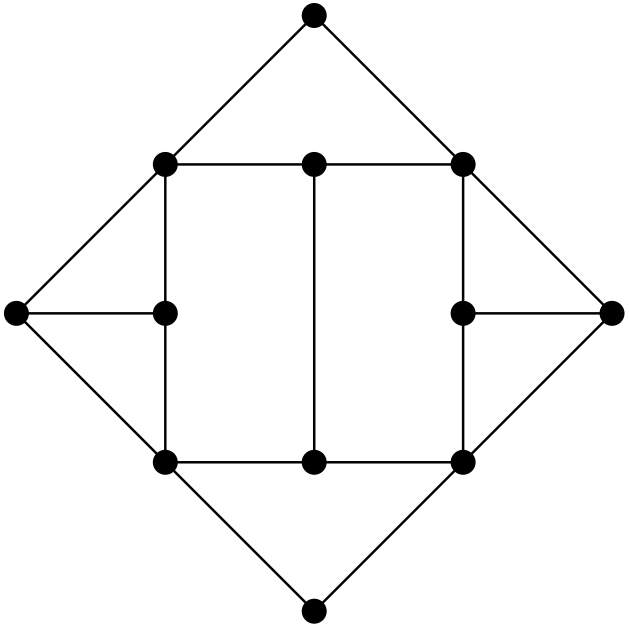
Although finding a Hamiltonian circuit may be hard, the following theorem says that for some graphs, knowing that a Hamiltonian circuit exists is relatively easy to assert.

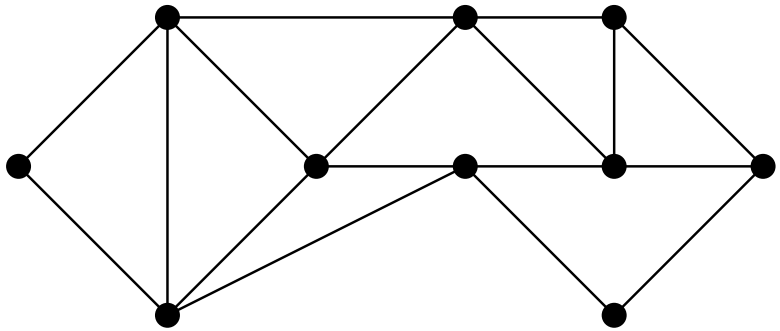
Theorem

If G is a simple connected graph with $n \geq 3$ vertices and if the degree of each vertex is greater than $n/2$, then G has a Hamiltonian circuit.

Finding Hamiltonian Circuits and Trails

Any Hamiltonian circuits or trails in these graphs?



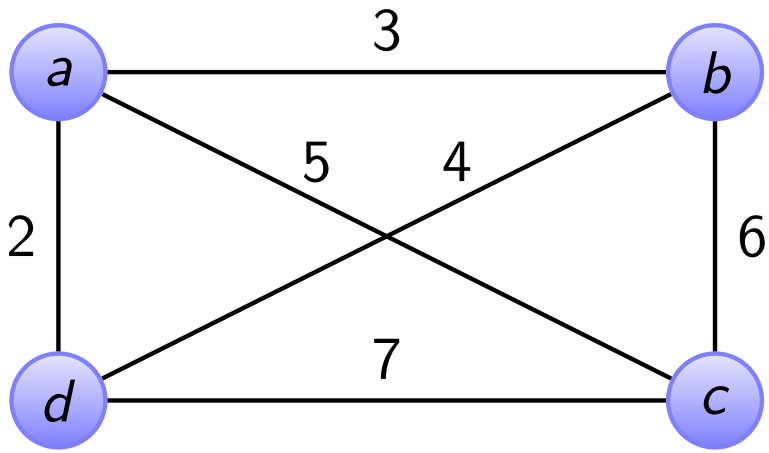


Weighted Graphs

Definition

A **weighted graph**, (V, E, w), is a graph (V, E) together with a weight function $w : E \to \mathbb{R}$. If $e \in E$, then $w(e)$ is the weight of edge e.

For example, the graph at the right is a weighted graph of K_4.



Often weights are associated with a *cost* or a *benefit* incurred when traversing the edge. Many important problems involve finding a trail, path, or circuit with minimum cost.

The Traveling Salesman Problem

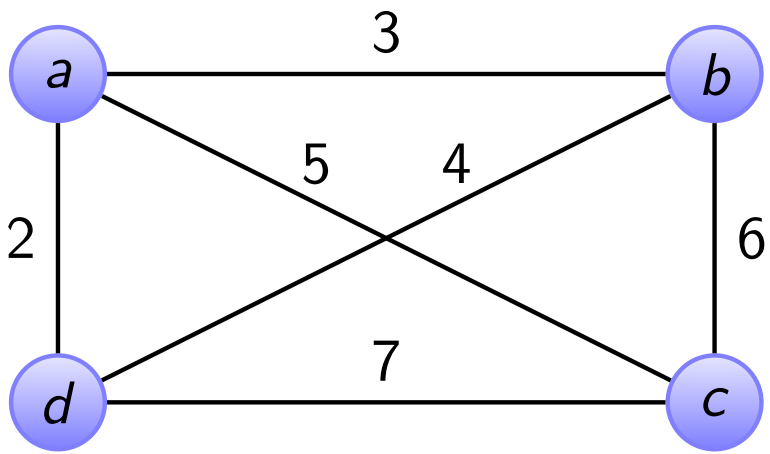
A very important problem is known as the **Traveling Salesman Problem** (TSP). Consider a salesman who must visit n different cities (or offices within a city). There is a cost associated with travel between each location he must visit and he's interested in completing his rounds as efficiently as possible based on some measure (least cost, or perhaps shortest time).

The TSP can be stated as: Given a connected, weighted graph G, find a Hamiltonian circuit in G of minimum total weight.

When the number of vertices in a graph is small, we can find every possible Hamiltonian circuit and just pick one with smallest weight.

The Traveling Salesman Problem

For example, consider all possible cycles (a circuit that only visits each vertex once, except the start/end vertex) in our weighted K_4.



cycle	weight
$a\,b\,c\,d\,a$	$3+6+7+2=18$
$a\,b\,d\,c\,a$	$3+4+7+5=19$
$a\,c\,b\,d\,a$	$5+6+4+2=17$
$a\,c\,d\,b\,a$	$5+7+4+3=19$
$a\,d\,b\,c\,a$	$2+4+6+5=17$
$a\,d\,c\,b\,a$	$2+7+6+3=18$

- There are six cycles – but half of these are reversals of the other half.
- The starting point is arbitrary – a given cycle will have the same cost regardless of starting point.

The Traveling Salesman Problem

Question: How many different Hamiltonian circuits does K_n have?

The Traveling Salesman Problem

Question: How many different Hamiltonian circuits does K_n have?

Answer: Every vertex in K_n is connected to every other vertex.

- There are $P(n, n) = n!$ ways to pick a starting vertex and choose a path that returns to the starting vertex.
- We must divide this by n since our count includes n identical cycles that differ only in starting point.
- We must divide by 2 since the forward and reverse of a cycle should be considered the same.

Thus, there are $\frac{n!}{2n} = \frac{(n-1)!}{2}$ different Hamiltonian circuits in K_n.

The Traveling Salesman Problem

Question: How many different Hamiltonian circuits does K_n have?

Answer: Every vertex in K_n is connected to every other vertex.

- There are $P(n, n) = n!$ ways to pick a starting vertex and choose a path that returns to the starting vertex.
- We must divide this by n since our count includes n identical cycles that differ only in starting point.
- We must divide by 2 since the forward and reverse of a cycle should be considered the same.

Thus, there are $\frac{n!}{2n} = \frac{(n-1)!}{2}$ different Hamiltonian circuits in K_n.

This is rather bad news...

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.
- When $n = 12$, however, it takes this computer 20 seconds to solve the TSP.

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.
- When $n = 12$, however, it takes this computer 20 seconds to solve the TSP.
- When $n = 15$ it will take over 12 hours. . .

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.
- When $n = 12$, however, it takes this computer 20 seconds to solve the TSP.
- When $n = 15$ it will take over 12 hours...
- and when $n = 20$ it would take over 1900 years.

The Traveling Salesman Problem

Suppose a computer can generate and check the cost of 10^6 Hamiltonian circuits in a graph every second. (For the sake of simplicity we assume that this time is independent of the number of vertices in the graph).

- When $n = 10$, this computer can solve the TSP in under a second.
- When $n = 12$, however, it takes this computer 20 seconds to solve the TSP.
- When $n = 15$ it will take over 12 hours...
- and when $n = 20$ it would take over 1900 years.

Granted, this is a brute-force approach to solving the TSP. The best known optimal algorithm has an operation count on the order of $n^2 2^n$ (i.e., $O(n^2 2^n)$), which means that when the number of vertices in the graph increases by one, it will take more than twice as long to find an optimal Hamiltonian circuit.

The Traveling Salesman Problem

This situation is bleak, but it is much better if we are willing to accept a *nearly optimal* solution to the TSP.

It is not unreasonable to assume that we are solving the TSP on a complete graph since edges do not necessarily indicate routes between locations but costs associated with traveling from one location to another.

Two simple algorithms we'll consider for the TSP on a complete graph are both **greedy** algorithms; they are multistep algorithms that make optimal choices at each step with the assumption that this will lead to a near optimal overall result.

In practice, these often work well but can produce far from optimal circuits in certain cases.

The Traveling Salesman Problem

Let G be a weighted graph based on K_n.

Vertex Greedy Algorithm (VGA)

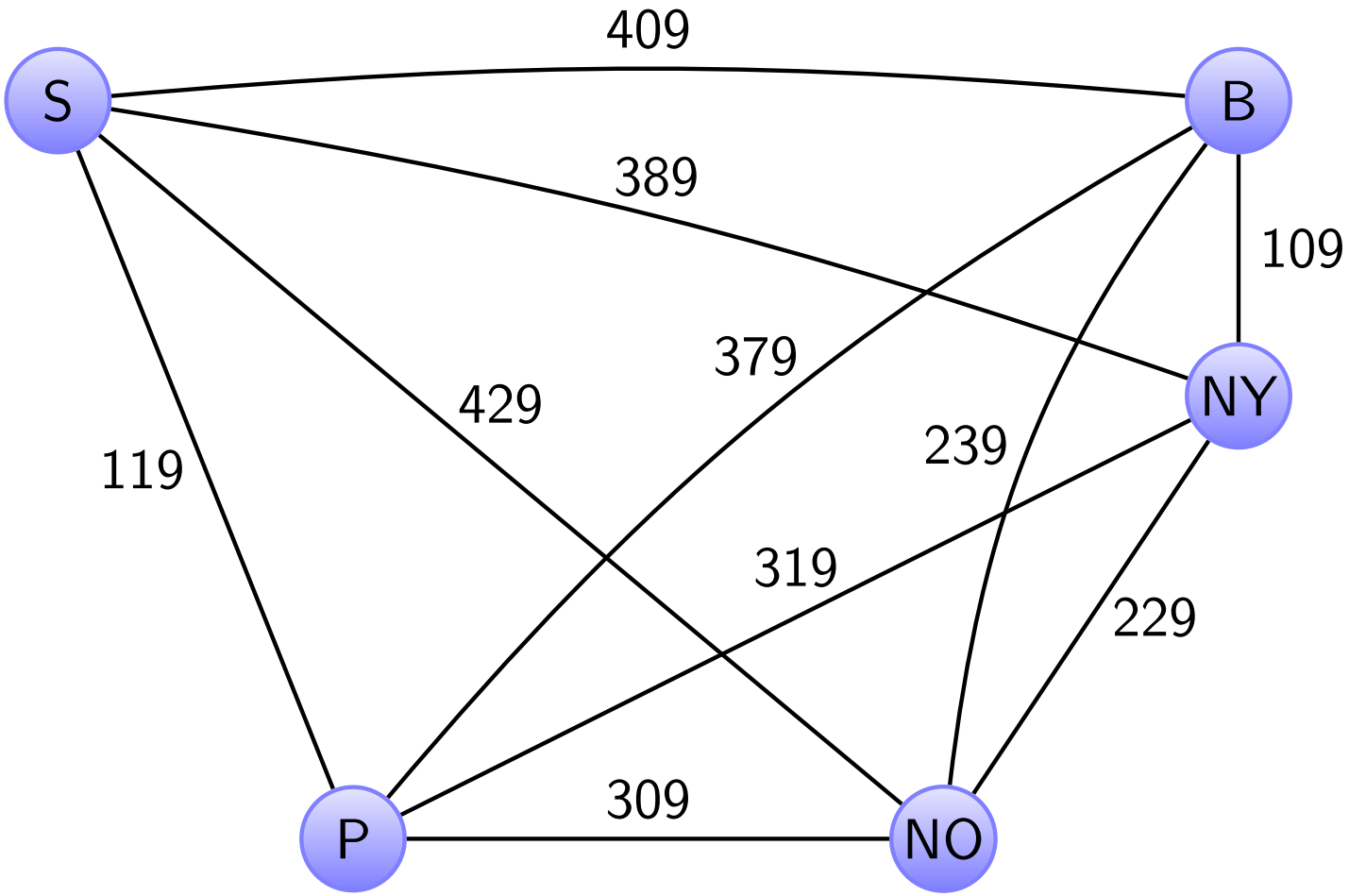
1. Identify starting vertex as v_1 and create set $V = \{v_1\}$.
2. For $i = 2$ to n:
   - Let v_i be an unvisited vertex v for which the edge from v_{i-1} to v has minimum weight.
   - $V = V \cup \{v_i\}$.
3. Set $v_{n+1} = v_1$.

Edge Greedy Algorithm (EGA)

1. Sort edges by weight.
2. Identify edge of minimum weight as e_1 and create set $V = \{v : e_1 \text{ is incident to } v\}$. Initialize $i = 2$.
3. While $|V| < n$:
   - Let e_i be an edge of minimum weight that does not create a cycle of length less than n or create a vertex of degree 3 in V.
   - $V = V \cup \{v : e_i \text{ incident to } v\}$.
   - $i = i + 1$.

The Traveling Salesman Problem

The following graph shows the cost of flying between Seattle (S), Phoenix (P), New Orleans (NO), New York (NY), and Boston (B).



The Traveling Salesman Problem

Example

Find near-optimal Hamiltonian cycles using the VGA and the EGA.

VGA

V	Cost
B	\$0
B, NY	\$109
B, NY, NO	\$338
B, NY, NO, P	\$647
B, NY, NO, P, S	\$766
B, NY, NO, P, S, B	\$1,175

EGA

e_i	Cost
(B,NY)	\$109
(S,P)	\$228
(NY,NO)	\$457
(NO,P)	\$766
(B,S)	\$1,175

In this case both algorithms find the same cycle; this isn't always true.

The Traveling Salesman Problem

- The graph in our last example is K_5, so there are $\frac{4!}{2} = 12$ Hamiltonian cycles. Checking all of them reveals that the optimal cycle is Boston, NY, Seattle, Phoenix, New Orleans, Boston with a cost of \$1,165.
- The ratio of the cost we found to the true optimal cost is

$$\frac{1175}{1165} \approx 1.0086$$

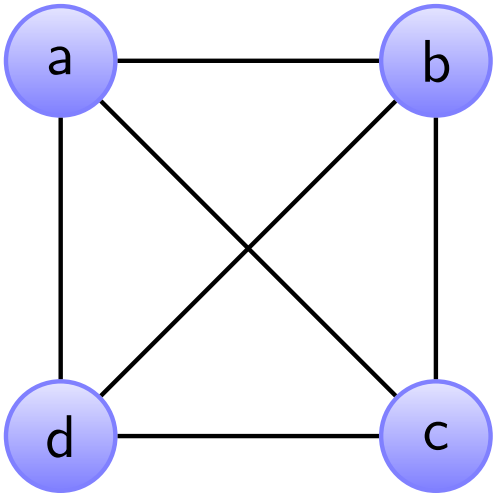
This means that the near-optimal cycle found incurs an extra cost of only 0.86%. In this case we'd probably consider the cycle we found to be acceptable.

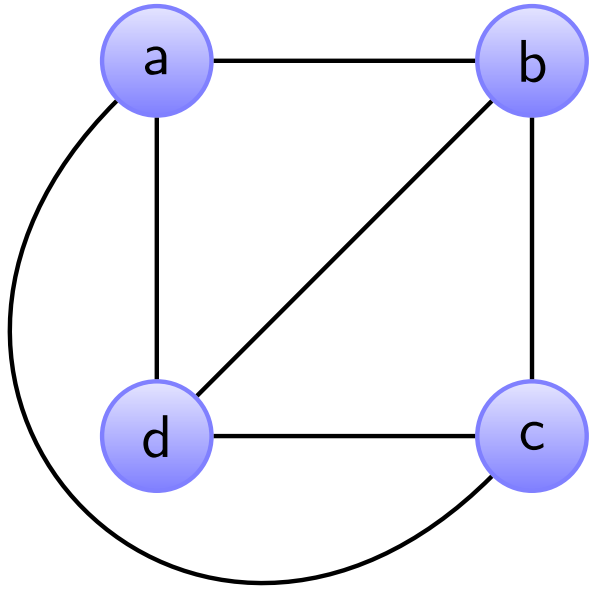
Planar Graphs

Definition

A simple connected graph is **planar** if it can be drawn in the plane without any edge crossings. Such a drawing is called an **embedding** of the graph in the plane.

The graph K_4, shown on the left, is planar because it can be drawn in the plane without any edge crossings.





Bipartite Graphs

Recall that:

Definition

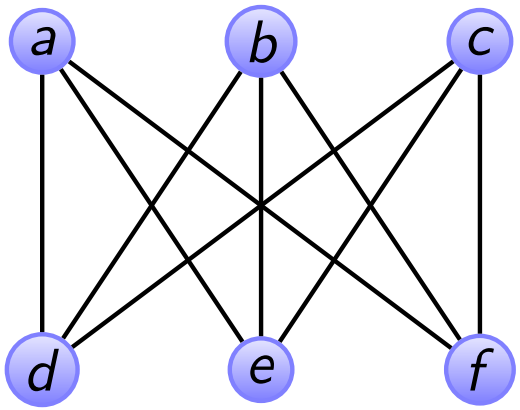
A graph is called **bipartite** if its set of nodes can be partitioned into two disjoint sets S_1 and S_2 so that every edge in the graph has one endpoint in S_1 and one endpoint in S_2.

Definition

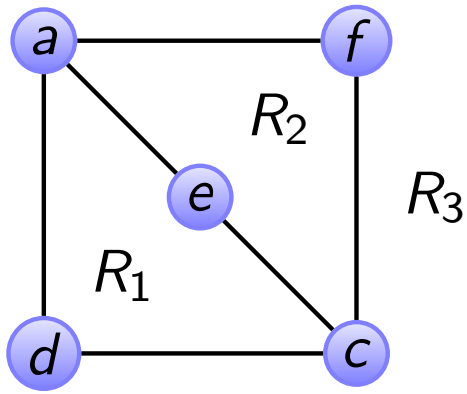
The **complete bipartite graph** on n, m nodes, denoted $K_{n,m}$, is the simple bipartite graph with nodes $S_1 = \{a_1, \ldots, a_n\}$ and $S_2 = \{b_1, \ldots, b_m\}$ and with edges connecting each node in S_1 to every node in S_2.

Is $K_{3,3}$ Planar?

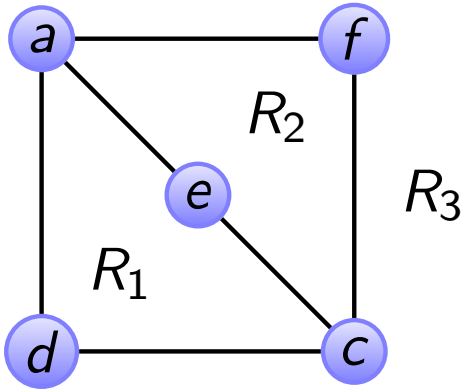
Is $K_{3,3}$ planar?



Suppose we start with a subgraph of $K_{3,3}$, which is planar. The following subgraph will partition the plane into three regions, R_1, R_2, and R_3.



Is $K_{3,3}$ Planar?



- If node b is put in R_1 it can be connected to d and e with no edge crossings but there is no way to connect it to f without a crossing.
- Similarly, if b is placed in R_2 we must cross an existing edge to get to d.
- Finally, if b is placed in R_3, we must cross an edge to get to e.

Similar investigations with other subgraphs will show that there is no way to draw $K_{3,3}$ without edge crossings, thus $K_{3,3}$ **is nonplanar**.

Euler's Formula

Euler showed that all planer representations of a graph partition the plane into the same number of regions.

Theorem (Euler's Formula)

If $G = (V, E)$ is a connected planar graph with r regions, v vertices, and e edges, then $v + r - e = 2$.

Proof.

Our proof is by induction. Starting with subgraph of G consisting a single vertex, we'll add edges and vertices (keeping the subgraphs connected) until we construct G. The basis step is

- R_1

Here $r = 1$, $v = 1$, $e = 0$ so $r + v = e - 2$ becomes $1 + 1 = 0 + 2$, which is true.

(Continued next slide)

Euler's Formula

Proof.

(Continued) For the inductive step, assume that we have a subgraph of G with $e = k$ edges and that $r + v = e + 2$ holds. We now draw the the $k + 1^{st}$ edge. There are two cases:

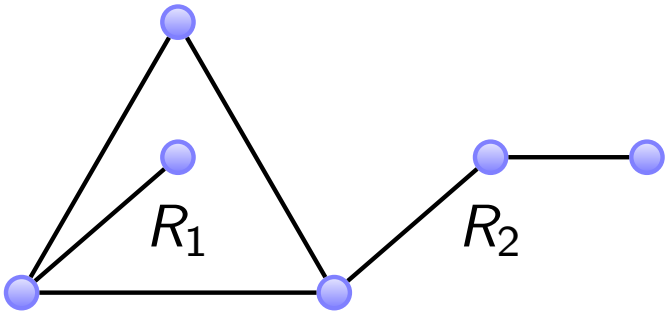
1. Both vertices the new edge is incident to are already on the graph. Since the subgraph is connected, this will create a new region. Thus both r and e increase by one so $r + v = e + 2$ is still true.
2. A new pendant vertex is introduced with the new edge. This does not increase r but does increase both e and v by one, so $r + v = e + 2$ continues to hold.

Since we can continue with these steps until we have constructed G, we conclude that $r + v = e + 2$ holds for any connected planar graph. □

Degrees of Regions

Definition

The **degree** of a region of a planar graph is the number of edge traversals required to trace the region's boundary.



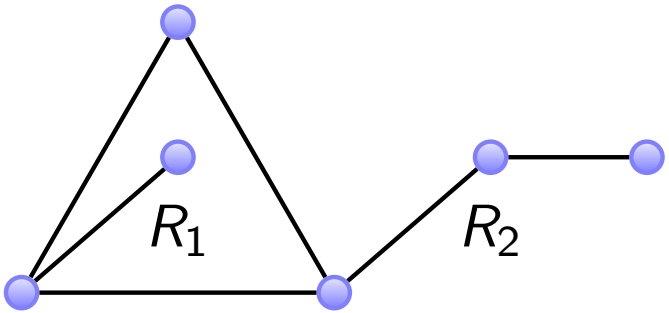
- $\deg(R_1) = 5$
- $\deg(R_2) = 7$.

Question: What is the relationship between the number of edges in a graph and the sum of the degrees of the regions formed by the graph?

Degrees of Regions

Definition

The **degree** of a region of a planar graph is the number of edge traversals required to trace the region's boundary.



- $\deg(R_1) = 5$
- $\deg(R_2) = 7$.

Question: What is the relationship between the number of edges in a graph and the sum of the degrees of the regions formed by the graph?

Answer: Each edge will be traversed twice, so

$$2e = \sum_{k=1}^{r} \deg(R_k).$$

A Simple Test for Nonplanarity

Question: Suppose G is a connected planar graph with at least three vertices. What is the minimum value of $\deg(R)$ for any region created by G?

A Simple Test for Nonplanarity

Question: Suppose G is a connected planar graph with at least three vertices. What is the minimum value of $\deg(R)$ for any region created by G?

Answer: $\deg(R_k) \geq 3$ for $k = 1, \ldots, r$.

A Simple Test for Nonplanarity

Question: Suppose G is a connected planar graph with at least three vertices. What is the minimum value of $\deg(R)$ for any region created by G?

Answer: $\deg(R_k) \geq 3$ for $k = 1, \ldots, r$.

In this case

$$2e = \sum_{k=1}^{r} \deg(R_k) \geq 3r = 3(e - v + 2)$$

so

$$\begin{aligned} 2e &\geq 3e - 3v + 6 \\ -e &\geq -3v + 6 \\ e &\leq 3v - 6. \end{aligned}$$

A Simple Test for Nonplanarity

This last result gives a useful test for nonplanarity.

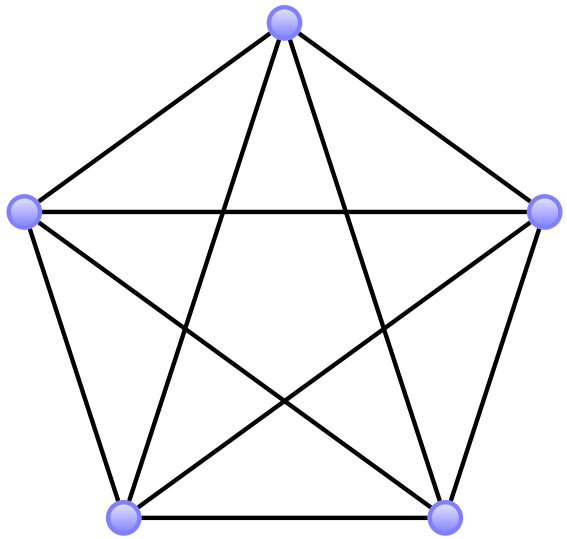
Theorem

Given a a connected planar graph G with $v \geq 3$ vertices and e edges, $e \leq 3v - 6$.

Thus, if $e > 3v - 6$ for a connected graph G with $v \geq 3$, we know that G is nonplanar.

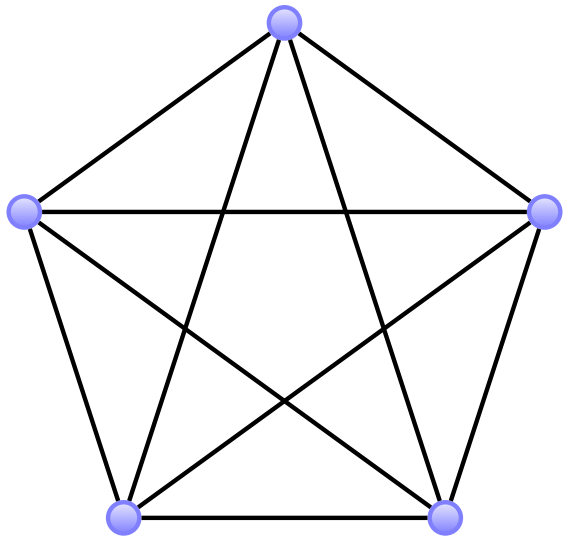
Is K_5 Planar?

Is K_5 planar?



Is K_5 Planar?

Is K_5 planar?



- Clearly $v = 5$ so $3v - 6 = 15 - 6 = 9$.
- We recall $e = C(5, 2) = 10$ so $e > 3v - 6$ and we can conclude that K_5 is nonplanar.

Kuratowski's Theorem

Definition

An **elementary subdivision** in a graph G replaces any edge $\{u, v\}$ with a new vertex w and two new edges $\{u, w\}$ and $\{w, v\}$.

Definition

Two graphs are **homeomorphic** if they can be obtained from the same graph via elementary subdivisions.

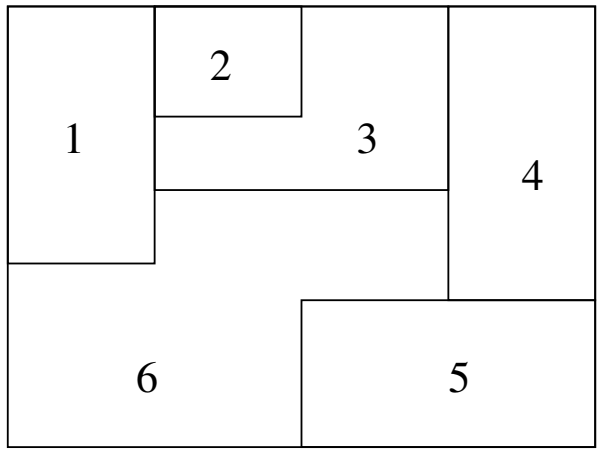
Theorem (Kuratowski's Theorem)

A graph is nonplanar if and only if it contains a subgraph homeomorphic to $K_{3,3}$ or K_5.

Coloring Graphs

Consider the "map" on the right.

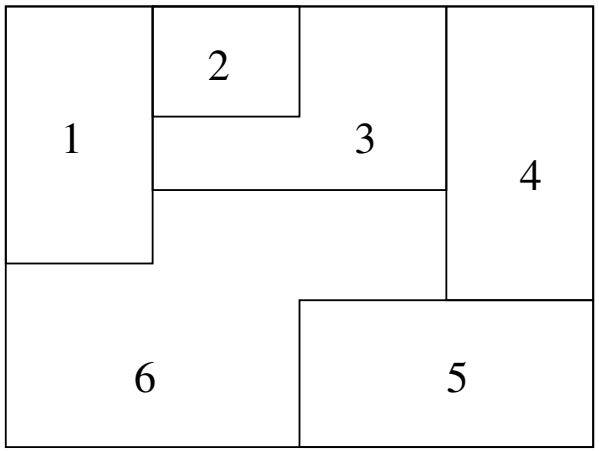
What is the fewest number of colors are needed to color this map so that no to adjacent regions have the same color?



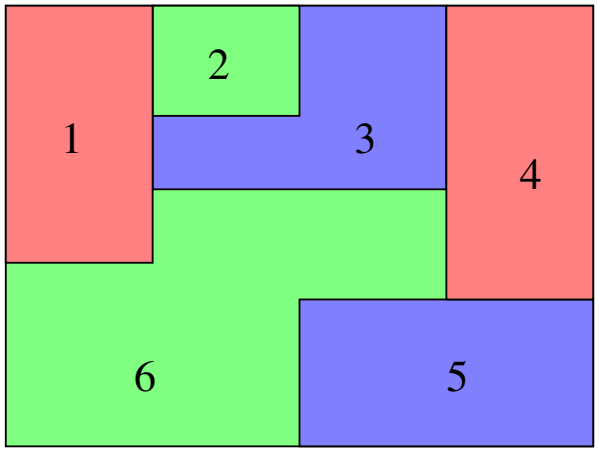
Coloring Graphs

Consider the "map" on the right.

What is the fewest number of colors are needed to color this map so that no to adjacent regions have the same color?



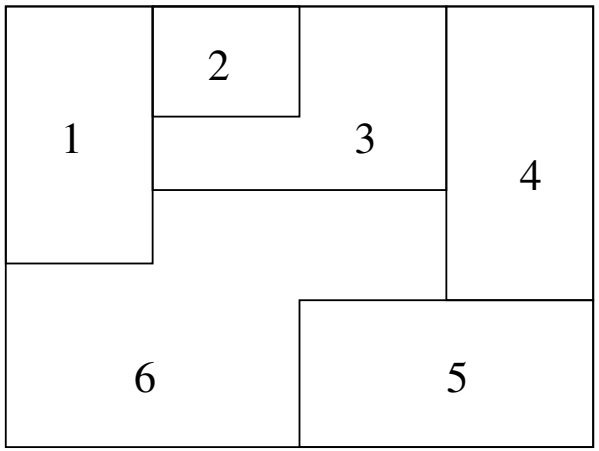
Two colors are not sufficient, but we can do it with three colors.



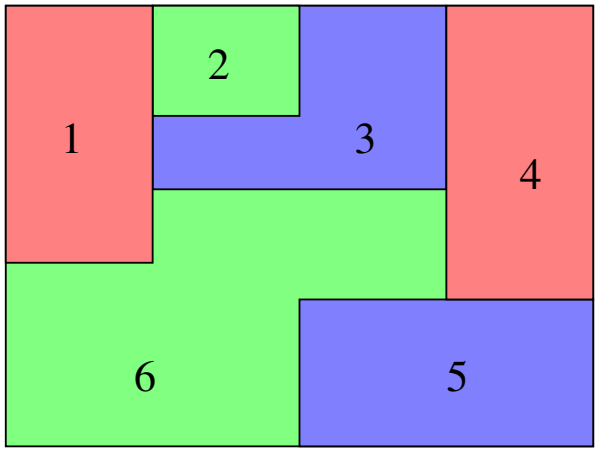
Coloring Graphs

Consider the "map" on the right.

What is the fewest number of colors are needed to color this map so that no to adjacent regions have the same color?



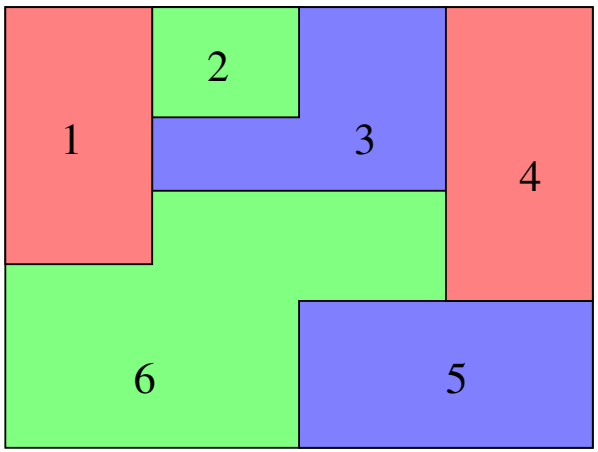
Two colors are not sufficient, but we can do it with three colors.

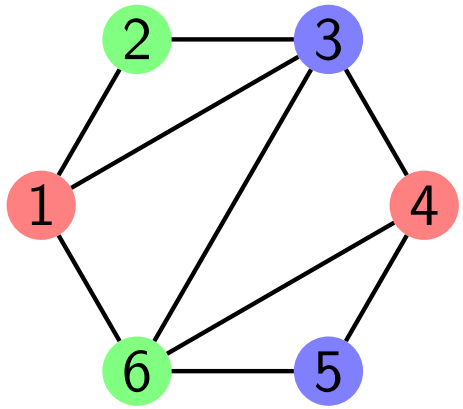


Question: What is the fewest number of colors that will be sufficient to color any map in the plane?

Dual Graphs of Maps

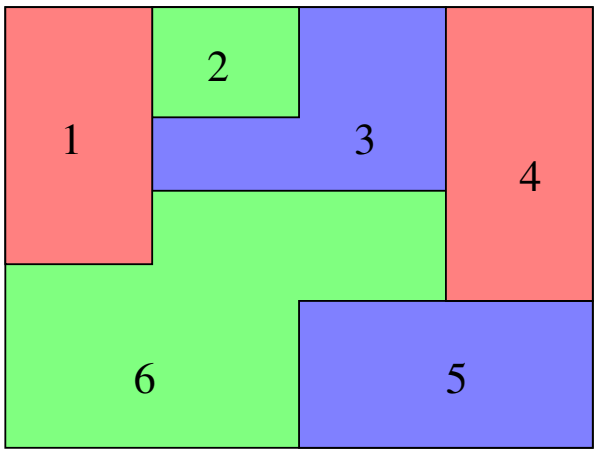
The **dual graph** of a map is a connected simple graph with one vertex corresponding to each region and an edge connecting two vertices if the corresponding regions on the map are adjacent.

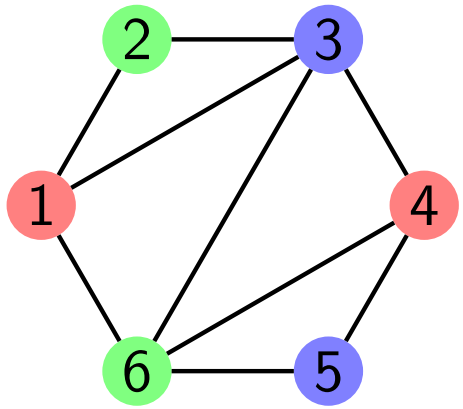




Dual Graphs of Maps

The **dual graph** of a map is a connected simple graph with one vertex corresponding to each region and an edge connecting two vertices if the corresponding regions on the map are adjacent.





Note that the dual graph of a map will always be planar.

Coloring a Graph

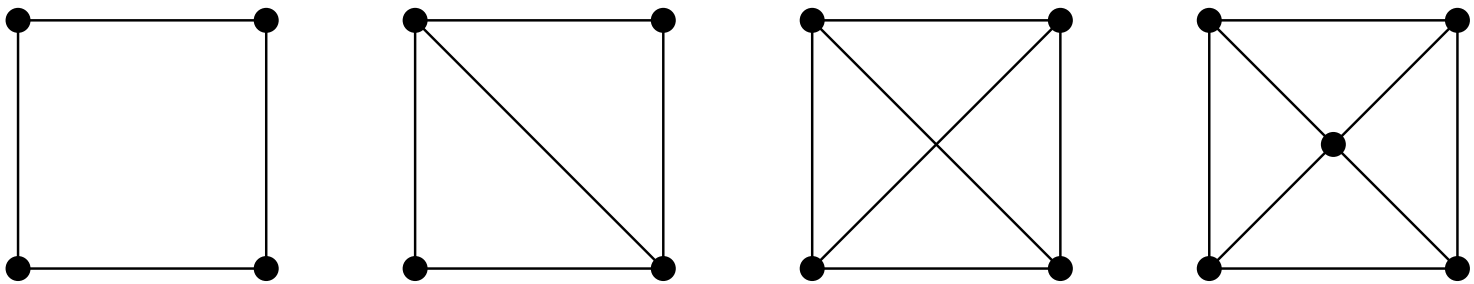
Definition

The **coloring** of a simple graph is the assignment of a color to each vertex of the graph so that no two adjacent vertices have the same color.

Definition

The **chromatic number** of a graph G is the least number of colors needed for a coloring of the graph and is denoted $\chi(G)$.

What is the chromatic number of these graphs?



The Four Color Theorem

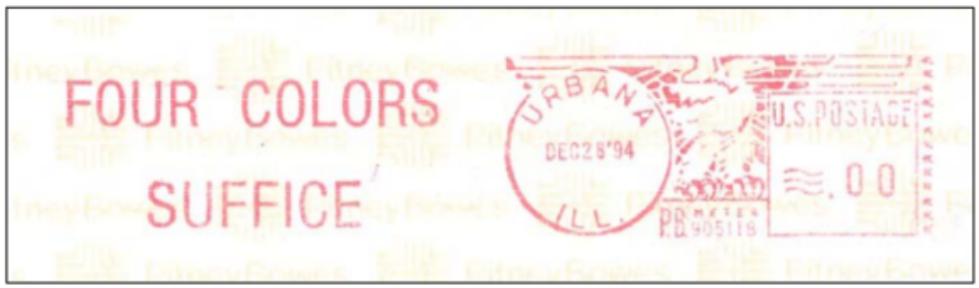
The problem of how many colors are needed to color a map is equivalent to asking "what is the largest chromatic number a planar graph can have?"

While first formulated in 1852, this question was not properly answered until 1976, establishing the following Theorem:

Theorem (Four Color Theorem)

The chromatic number of a planar graph is no greater than four.

- In that year Appel and Haken presented a proof by cases of the Four Color Theorem that used a computer to check about 1800 separate cases.
- Generally accepted as the first use of a computer to prove a theorem.



Postmark from 1994 commemorating the proof of the Four Color Theorem.

Chromatic Number

What is the chromatic number of the following graphs?

- K_n:
- $K_{m,n}$:
- C_n (the cycle graph with n vertices):

Chromatic Number

What is the chromatic number of the following graphs?

- K_n: Since each vertex will need its own color, $\chi(K_n) = n$.
- $K_{m,n}$:
- C_n (the cycle graph with n vertices):

Chromatic Number

What is the chromatic number of the following graphs?

- K_n: Since each vertex will need its own color, $\chi(K_n) = n$.
- $K_{m,n}$: The vertices of $K_{m,n}$ are partitioned into two sets with each edge connecting a vertex in one set to a vertex in the other set. Thus, only two colors are needed, so $\chi(K_{m,n}) = 2$.
- C_n (the cycle graph with n vertices):

Chromatic Number

What is the chromatic number of the following graphs?

- K_n: Since each vertex will need its own color, $\chi(K_n) = n$.

- $K_{m,n}$: The vertices of $K_{m,n}$ are partitioned into two sets with each edge connecting a vertex in one set to a vertex in the other set. Thus, only two colors are needed, so $\chi(K_{m,n}) = 2$.

- C_n (the cycle graph with n vertices): The chromatic number depends on the parity of n:
  - $\chi(C_n) = 1$ if $n = 1$
  - $\chi(C_n) = 2$ if n is even
  - $\chi(C_n) = 3$ if n is odd and $n > 1$

Example

Suppose a summer institute offers seven courses. Students can take more than one course.
The following lists show all pairings of courses that have at least one student in common:

Course 1 : 2, 3, 4, 7

Course 2 : 3, 4, 5, 7

Course 3 : 4, 6, 7

Course 4 : 5, 6

Course 5 : 6, 7

Course 6 : 7

Find the fewest number of final exam slots that are needed to avoid any conflicts.

Example

To solve this problem we construct a graph where each vertex represents a course and an edge between vertices means the corresponding courses have a student in common. We then color the graph to find its chromatic number.

Course 1 : 2, 3, 4, 7

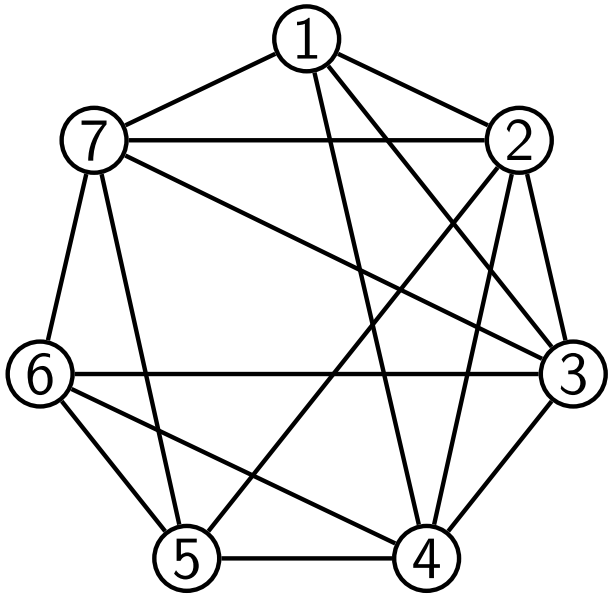
Course 2 : 3, 4, 5, 7

Course 3 : 4, 6, 7

Course 4 : 5, 6

Course 5 : 6, 7

Course 6 : 7



Example

To solve this problem we construct a graph where each vertex represents a course and an edge between vertices means the corresponding courses have a student in common. We then color the graph to find its chromatic number.

Course 1 : 2, 3, 4, 7

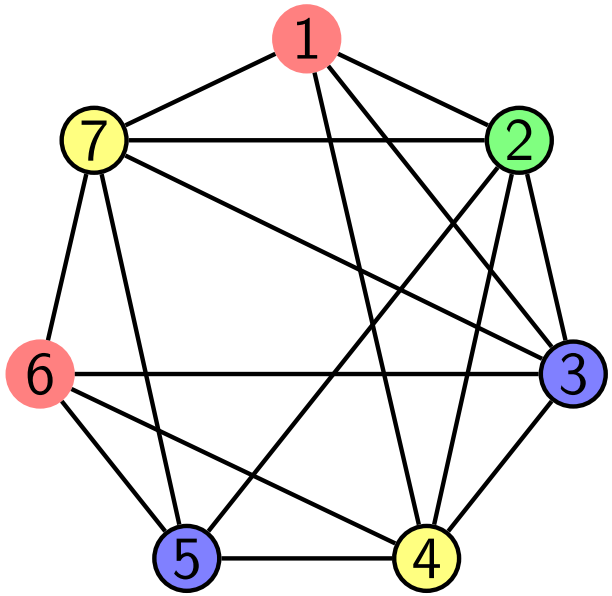
Course 2 : 3, 4, 5, 7

Course 3 : 4, 6, 7

Course 4 : 5, 6

Course 5 : 6, 7

Course 6 : 7



Since the chromatic number is 4, we conclude that four different final exam periods will be sufficient.