Exam 1 Review

For Math 154 Spring 2020

Basic Graph Concepts (Ch 1.1)

- What is a graph?
- Drawing graphs
- Basic terminology
- Basic types of graphs
- Walks, Paths, and Connectivity

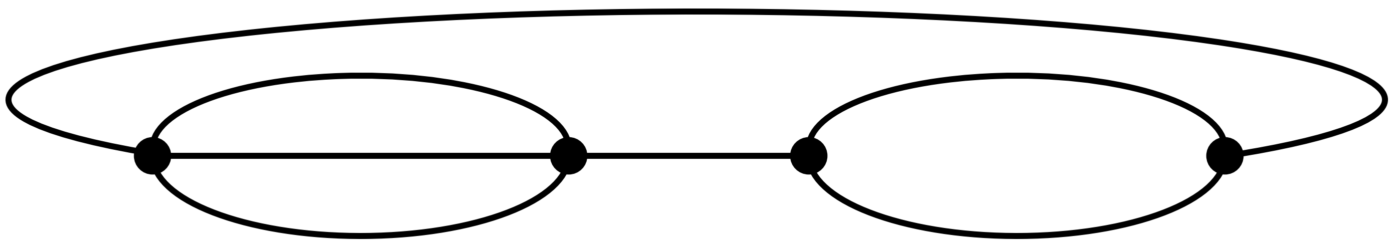
Graph Definition

Definition: A *graph* G = (V,E) consists of two things:

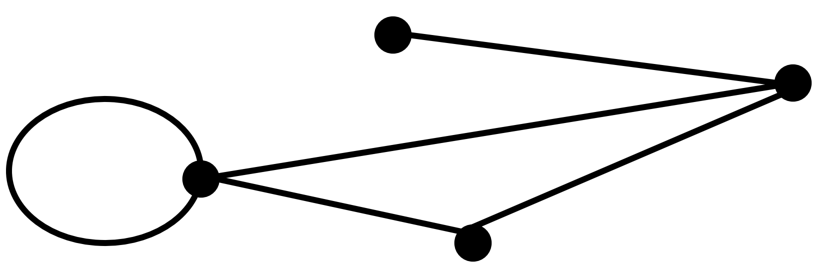
- A collection V of *vertices*, or objects to be connected.
- A collection E of *edges*, each of which connects a pair of vertices.

Other Types of Graphs I

A *mutligraph* can have multiple edges between the same pair of vertices.



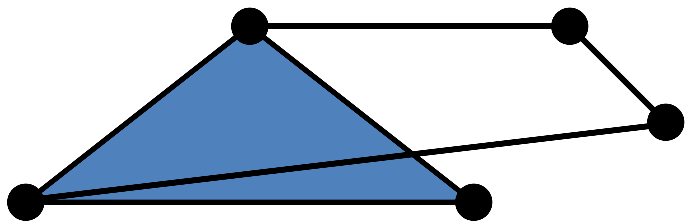
A *pseduograph* can have loops, edges connecting a vertex to itself.



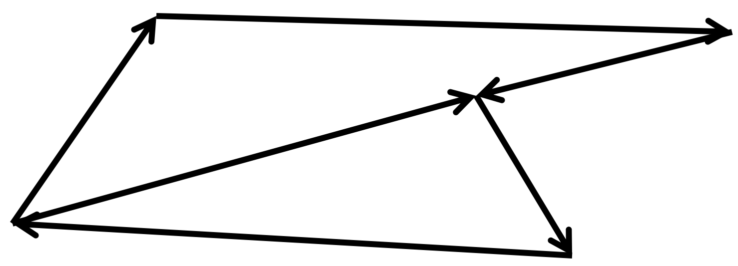
A graph is called *simple* if it has neither.

Other Types of Graphs II

A *hypergraph* can have edges that connect more than two vertices.

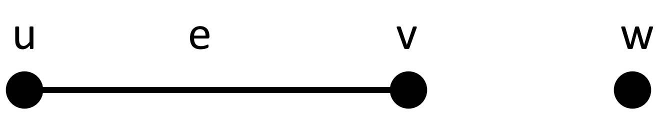


A *directed graph* has edges that only point in one direction



Graph Terminology I

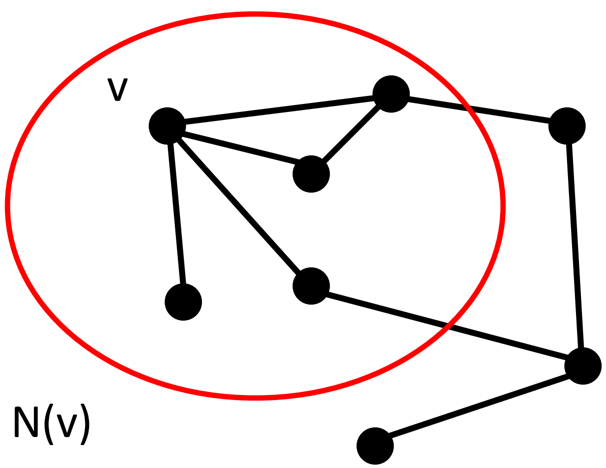
- Two vertices u and v are *adjacent* if there is an edge connecting them.
- A vertex v is *incident* on an edge e (or is an *endpoint* of e) if v is one of the vertices e connects.



u is adjacent to v
v is non-adjacent to w
u is incident on e
w is not incident on e

Graph Terminology II

- The *neighborhood* of a vertex v (denoted N(v)) is the set of vertices adjacent to v along with v.
- The *degree* of v (denoted d(v)) is the number of vertices adjacent to v.

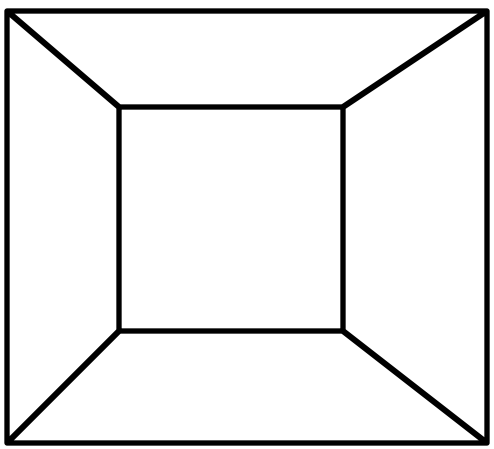


d(v) = 4

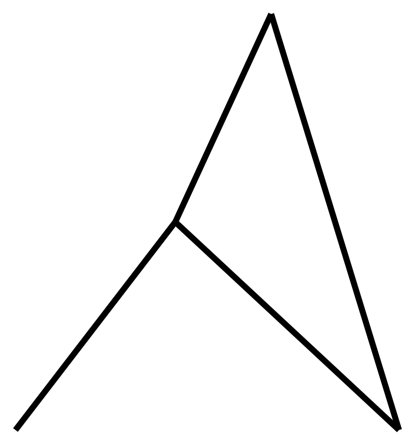
Graph Terminology III

- A graph is *d-regular* if all vertices have degree d. It is *regular* if it is d-regular for some d.

This graph is 3-regular



This graph is not regular



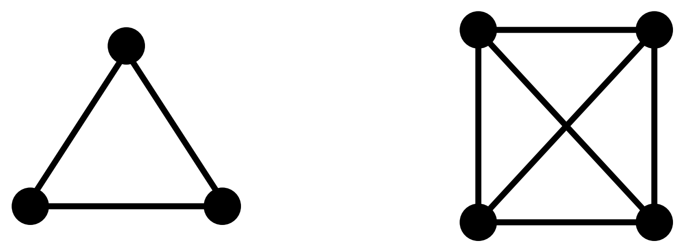
The Handshake Lemma

(Theorem 1.1) For any graph $G = (V, E)$,

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

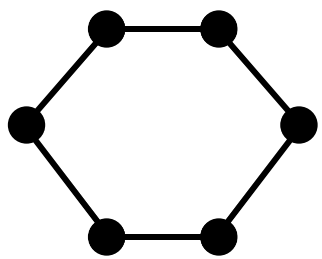
Examples of Graphs I

A *complete graph* on n vertices (denoted K_n) is a graph with n vertices and an edge between every pair of them

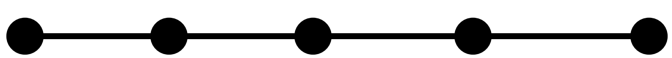


Examples of Graphs II

A *cycle* on n vertices (denoted C_n) is a graph with n vertices connected in a loop.

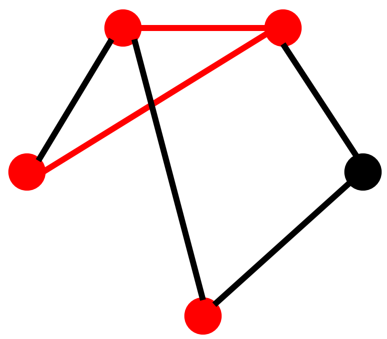


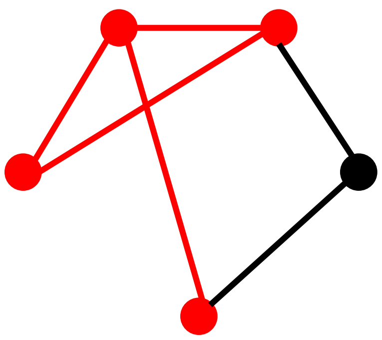
A *path* on n vertices (denoted P_n) is a graph with n vertices connected in a chain.



Examples of Graphs III

A graph H is a *subgraph* of G if V(H) ⊂ V(G) and E(H) ⊂ E(G).

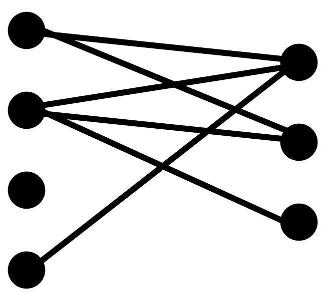


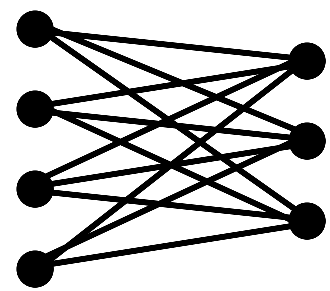


A subgraph H is an *induced subgraph* if it contains *all* the edges of G connecting two vertices in V(H).

Examples of Graphs IV

A *bipartite graph* is a graph whose vertices can be split into two parts where all edges connect one part to the other.





A *complete bipartite graph* (denoted $K_{n,m}$) has an edge connecting every element of one part (of size n) to every element of the other (of size m).

Walk Definitions

A *walk* in a graph G is a sequence of vertices $v_1, v_2, \ldots, v_n$ where for each i, v_i and v_{i+1} are connected by an edge.

Types of Walks

A walk whose edges are distinct is called a *trail*.

A walk whose vertices are distinct is called a *path*.

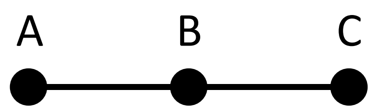
A *circuit* is a trail that starts and ends at the same vertex.

A *cycle* is a path plus an additional edge connecting the ends.

Induction on Length

The *length* of a walk is the number of edges in that walk.

For example, the walk ABCBA below is length 4.



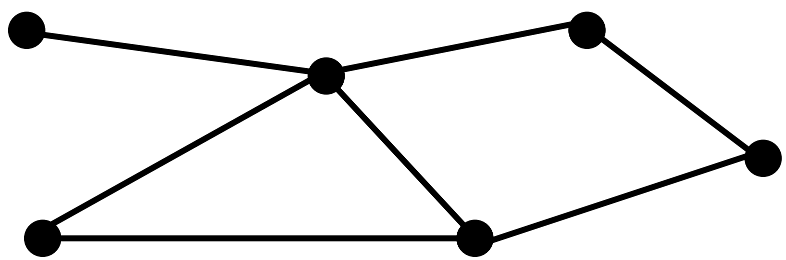
The Importance of Paths and Cycles

Lemma (Theorem 1.2): In a graph G every walk from vertex u to vertex v (a u-v walk) contains a u-v path (by removing some of the edges).

Similarly, every circuit contains a cycle.

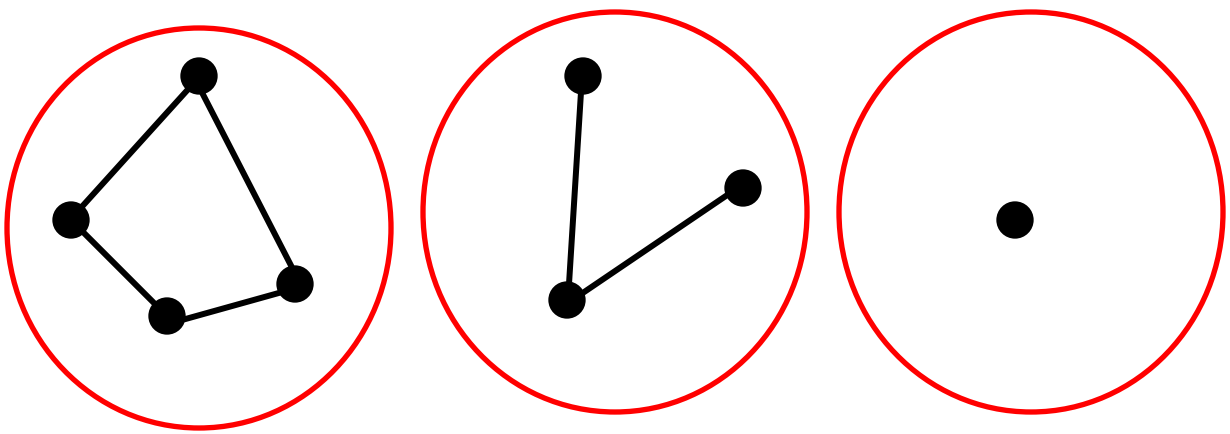
Connectivity

A graph G is *connected* if for any two vertices, u and v there is a u-v path in G.



Connected Components

Theorem: Any graph G can be uniquely partitioned into *connected components*, where each component is a connected subgraph and no two components have any edges between them.



Classification of Bipartite Graphs

G is bipartite if you can color vertices black & white so that all edges connect a black vertex to a white vertex.

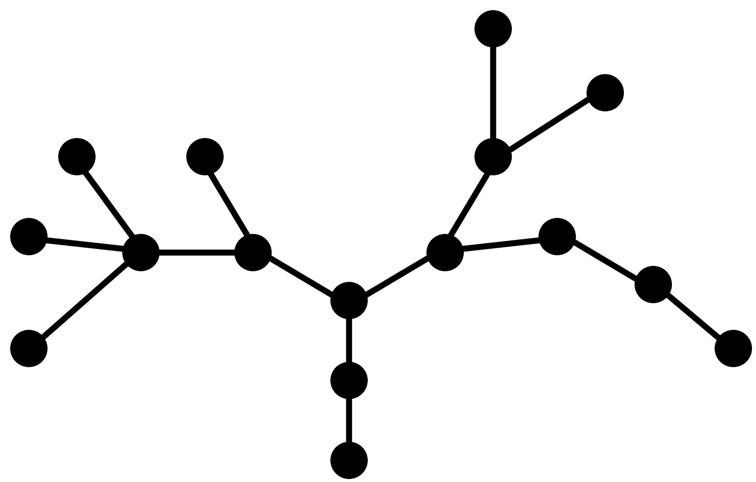
Theorem (Theorem 1.3): A graph G is bipartite if and only if it has no cycles of odd length.

Trees (Chapter 1.3)

- Definition and motivation
- Basic Properties
- Spanning Trees
- Counting Problems

Trees

Definition: A *tree* is a connected graph with no cycles. A *forest* is a graph where each connected component is a tree.



A Lemma

Lemma: Let T be a tree with vertices u and v. There exists a *unique* u-v path in T.

Edge Count

Theorem (1.10): Any tree with n vertices has exactly n-1 edges.

Idea: Look at connected components.

Lemma: Any graph $G = (V,E)$ with no cycles has $|V| - |E|$ connected components.

Leaves

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

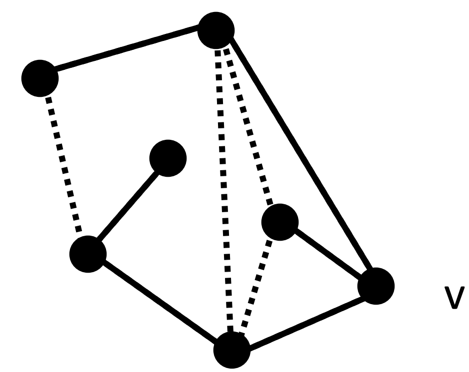
Lemma (Thrm 1.14): Any tree on $n > 1$ vertices has at least two leaves.

Spanning Trees

Definition: In a graph G a *Spanning Tree* is a subgraph T of G that is a tree using all of the vertices of G.

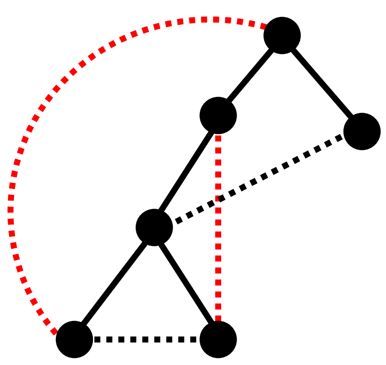
Breadth First Search Tree

- Start at a base vertex v
- Connect v to all its neighbors
- Connect them to all their neighbors (without creating cycles)
- Repeat until you've reached all vertices



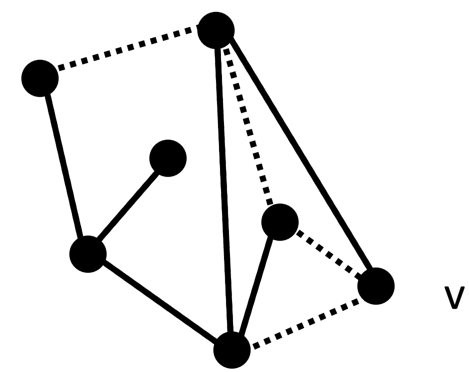
Breadth First Search Properties

- Finds *shortest* paths from v to other vertices.
  - d^{th} round of edges finds all vertices reachable from v with paths of length d.
- No edges in G provide shortcuts from a vertex to its descendants further down.



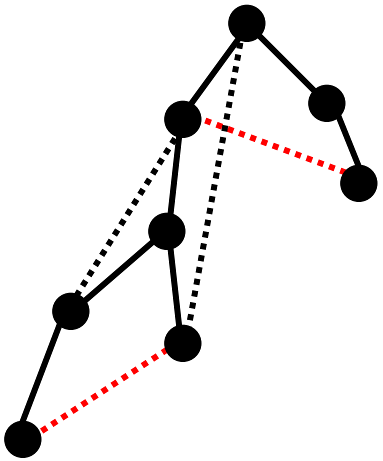
Depth First Search Tree

- Start at a base vertex v
- Follow path from v until cannot extend anymore
- Backtrack until new branch
- Repeat backtrack/extend until nothing else to do



Depth First Search Tree Properties

- G has no extra edges that cross between different branches of the tree.



Minimum Spanning Tree

Definition: For a graph G with edge weights, a *Minimum Spanning Tree* is a spanning tree of G whose sum of edge weights is as small as possible.

How do you find a MST?

Kruskal's Algorithm:

- Repeatedly add lightest edge that does not create a cycle

Prim’s Algorithm

Another way to find MST:

- Start at base vertex
- Repeatedly add cheapest new edge connected base vertex to something new

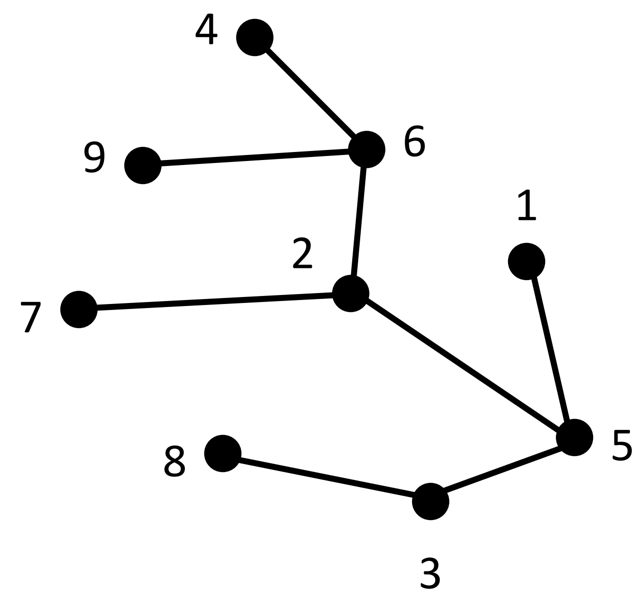
Cayley's Theorem

Theorem (1.18): There are n^{n-2} labeled trees of order n.

Proof idea: Find a bijection between trees and sequences of n-2 numbers from 1,2,..,n

How to get a List from a Tree

- Take lowest labeled leaf, v
- Record label of v's neighbor
- Remove v from G
- Repeat until G has only 2 vertices



5, 6, 2, 3, 5, 2, 6

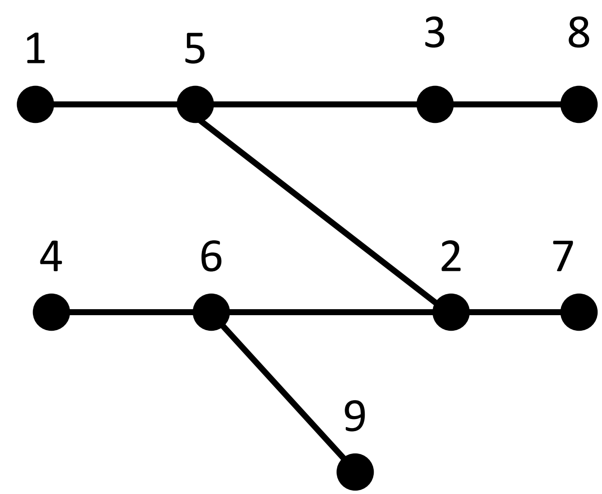
How to find the tree

- Find missing numbers. These are the leaves.
- Smallest missing number is v.
- Connect v to first element of list.
- Remove v from available numbers and first element of list
- Repeat until list gone
- Connect remaining elements

5, 6, 2, 3, 5, 2, 6

Missing: 1, 4, 7, 8, 9

3, 5, 2,



1, 2, 3, 4, 5, 6, 7, 8, 9

Paths and Cycles (Ch 1.4)

- Eulerian Circuits
  - Definition
  - Classification of Eulerian graphs
  - Algorithms
- Hamiltonian cycles
  - Definition
  - Hardness
  - Some conditions

Definitions

An *Eulerian circuit* is a circuit that uses every edge of a graph exactly once.

An *Eulerian trail* similarly uses each edge exactly once, but does not start and end at the same vertex.

A graph is *Eulerian* if it contains an Eulerian circuit and *semi-Eulerian* if it contains an Eulerian trail.

Conclusion

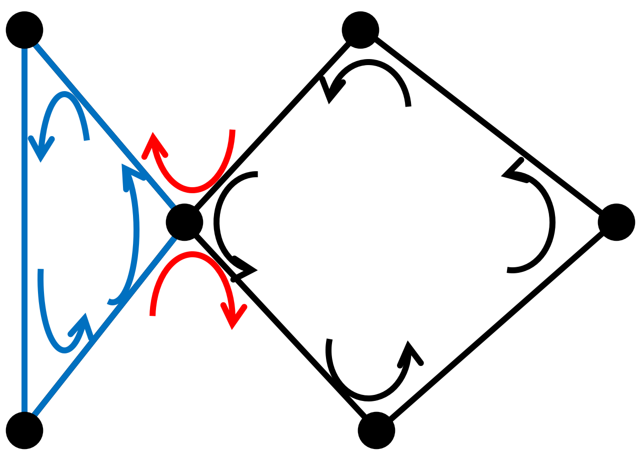
Theorem (1.20): A finite, connected graph G is Eulerian if and only if all vertices have even degree.

Constructing a Circuit

- If every degree is even you can construct a circuit by starting at v and following new edges until you get stuck (must be at v).
- Repeat on remaining edges, to partition edges of G into circuits.

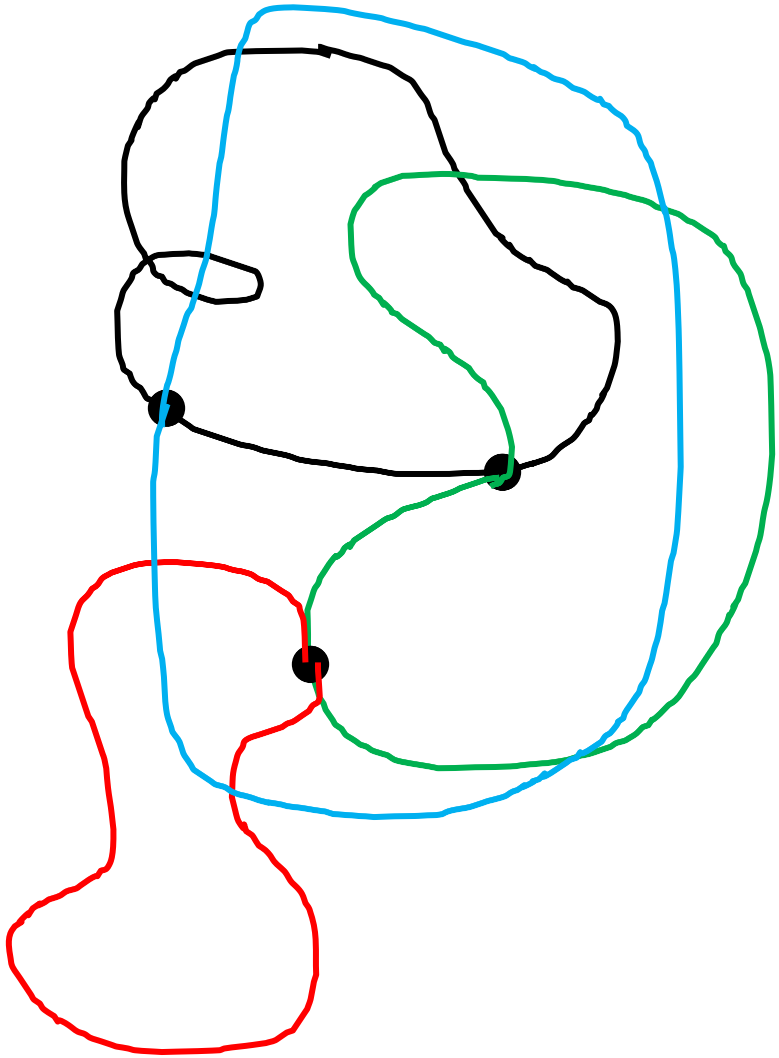
Combining Circuits

- Have two circuits that share a vertex.
- Turn them into one big circuit.



Final Algorithm

- Find a circuit.
- If all of G -> done.
- Otherwise, find v on circuit with unused edge.
- Find additional circuit through v.
- Merge with existing circuit.
- Repeat



Semi-Eulerian Graphs

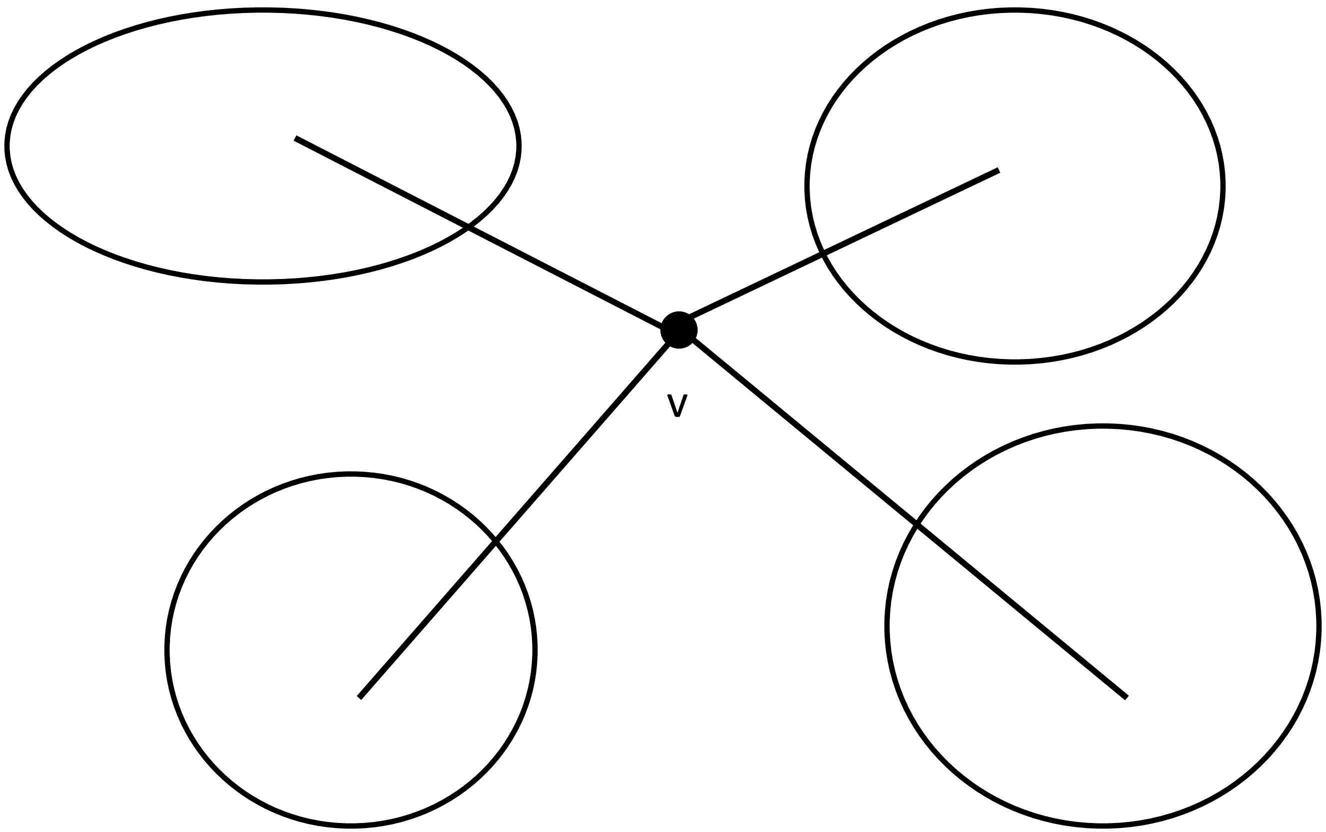
Theorem: A finite, connected graph G is semi-Eulerian if and only if it has exactly two vertices of odd degree. Furthermore, these vertices will be the endpoints of any Eulerian trial.

New Algorithm

Build Eulerian circuit/trail by starting at correct vertex, follow *any* edge that isn't a bridge.

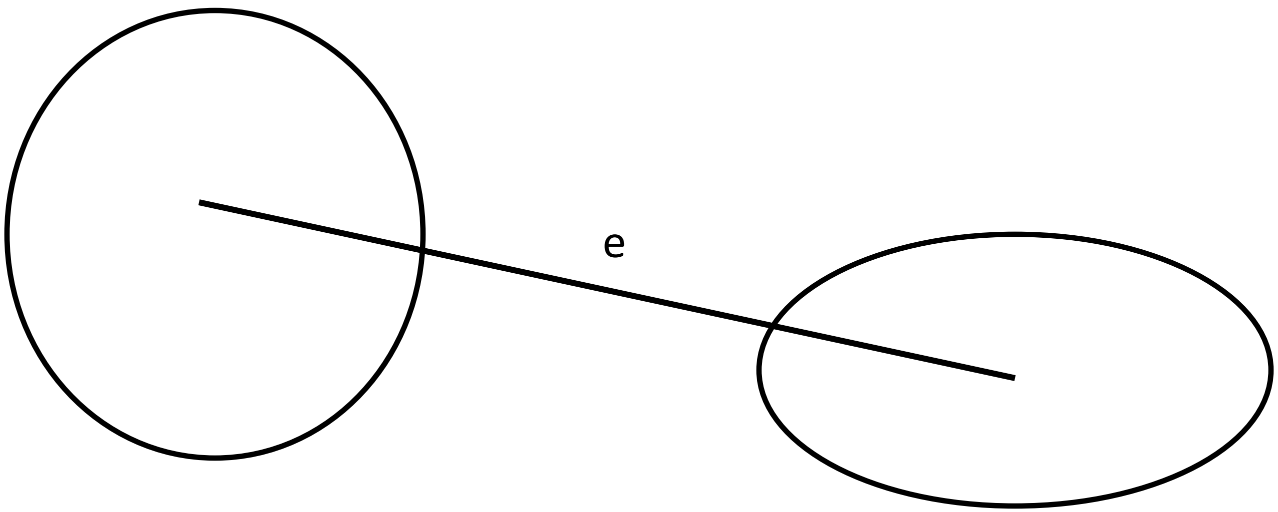
The Bad Case

This is what we want to avoid, is it possible?



Bridges and Odd Degrees

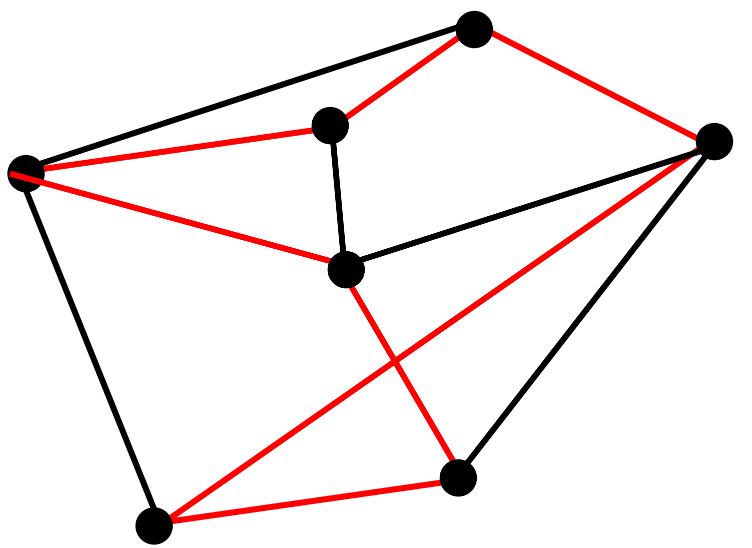
Lemma: If e is a bridge in a finite graph G, then there is at least one vertex of odd degree on each side of it.



Hamiltonian Graphs

Definition: A *Hamiltonian path/cycle* in a graph G is a path/cycle that uses every vertex of G exactly once.

A graph is *Hamiltonian* if it has a Hamiltonian cycle



Comparison

<u>Eulerian Graphs:</u>

- Is there a circuit that uses every *edge* exactly once?
- Easy characterization based on number of odd degree vertices
- Simple algorithms to construct

<u>Hamiltonian Graphs:</u>

- Is there a circuit that uses every *vertex* exactly once?
- No characterization
- NP-Hard to construct

Focus on partial characterizations.

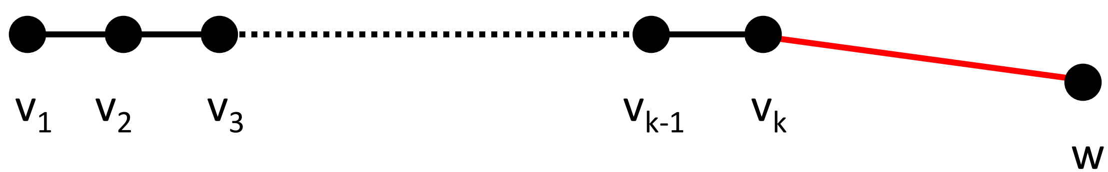
Minimum Degrees

Theorem (1.22): If a graph G on $n > 2$ vertices has minimum degree at least $n/2$, it is Hamiltonian.

Longest Path

Back to our main proof. Let $\delta(G) \geq n/2$.

Let $v_1, v_2, \ldots, v_k$ be the *longest* path in G.

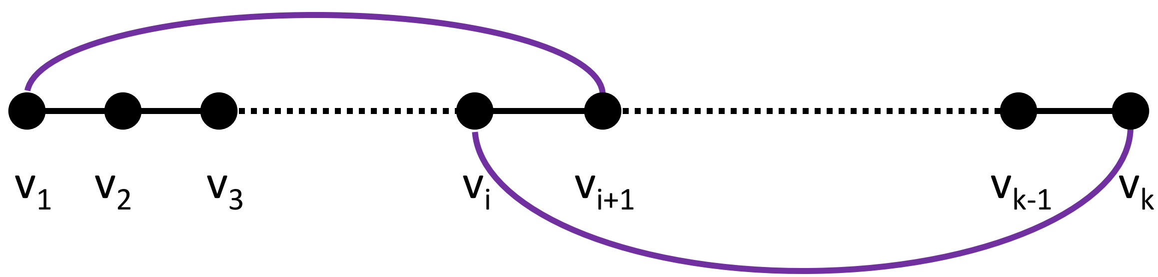


Note that v_1 and v_k cannot connect to vertices outside of this path.

Claim: G has a cycle of length k.

Counting

- There are ≥ n/2 i's so that v_k connects to v_i.
- There are ≥ n/2 i's so that v_1 connect to v_{i+1}.
- There are only k-1 < n total i's.
- The two lists must have some index in common.

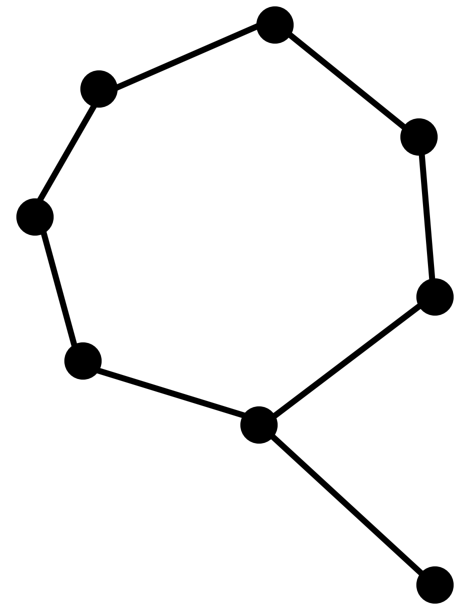


Cycle

Have a cycle of length k.

G is connected so either:

- k=n.
  - Cycle is Hamiltonian
- k<n
  - Some vertex in cycle connects to some other vertex.
  - We have a longer path.



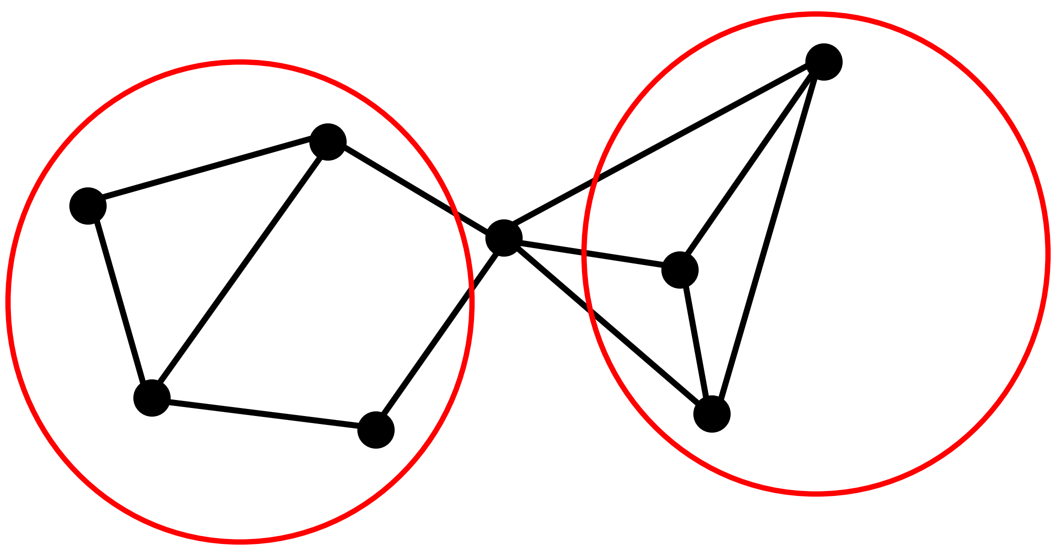
Structure of Connected Graphs (Verstraete Ch 4)

- Blocks and cut vertices
- Block decomposition of graphs
- Ear decompositions
- Theta-less graphs
- Menger’s Theorem

Cut Vertices

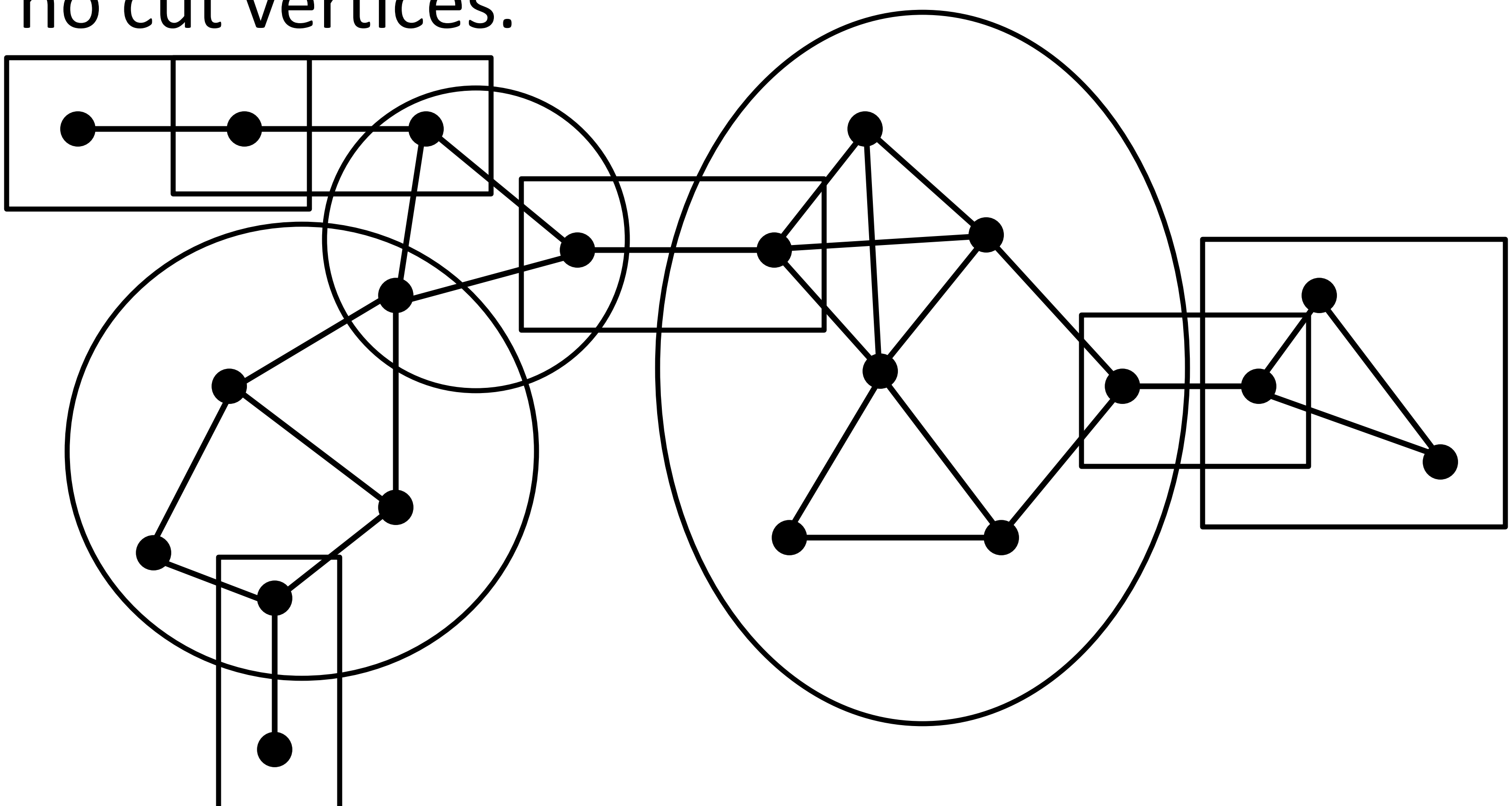
How do you break up a connected graph?

Definition: A *cut vertex* in a connected graph G is a vertex v so that G-v is disconnected.



Blocks

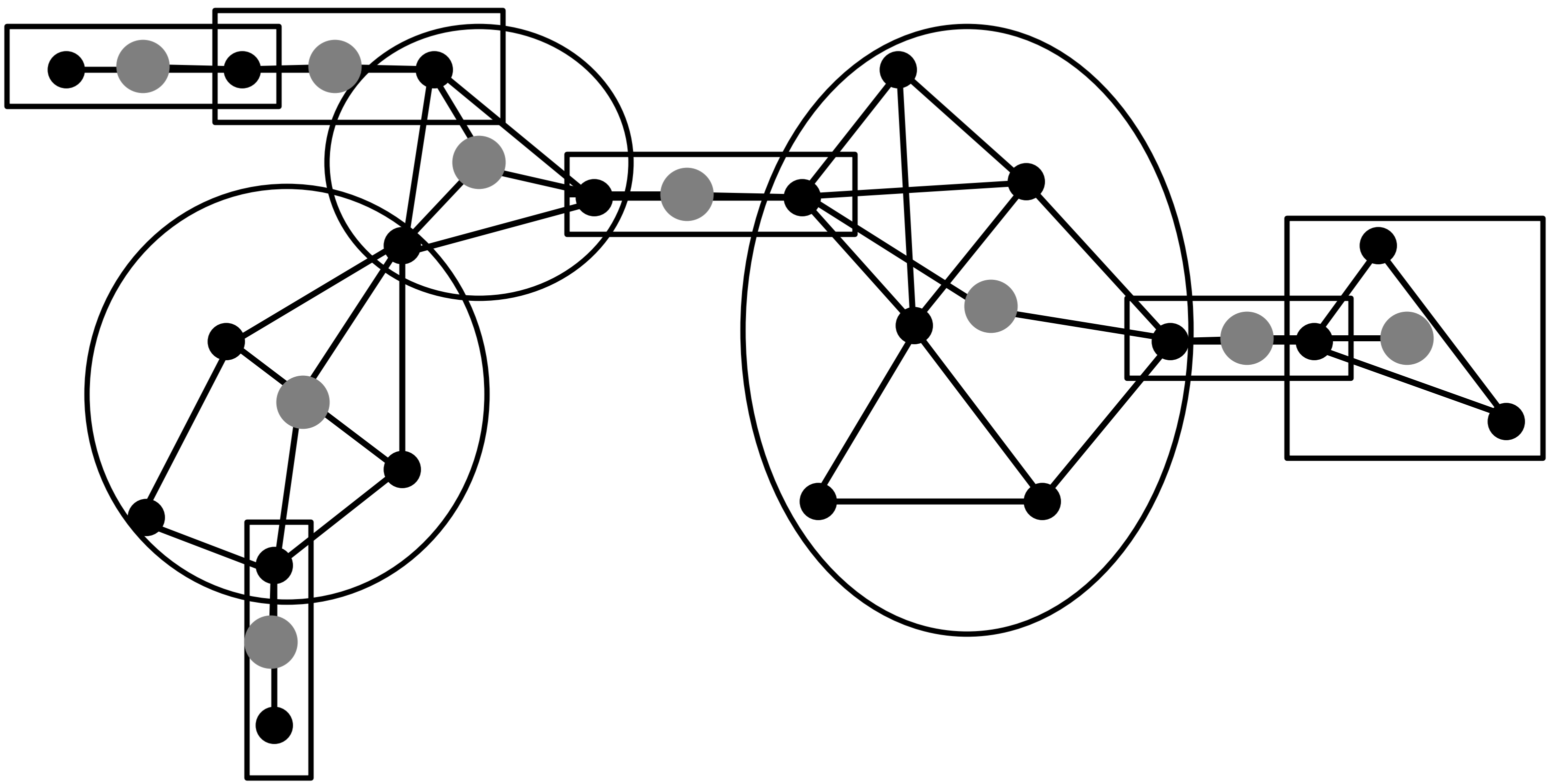
<u>Definition:</u> A *block* is a maximal subgraph with no cut vertices.



Intersection of Blocks

Lemma: The intersection of two blocks is either empty of consists of a single cut vertex.

Block Graph

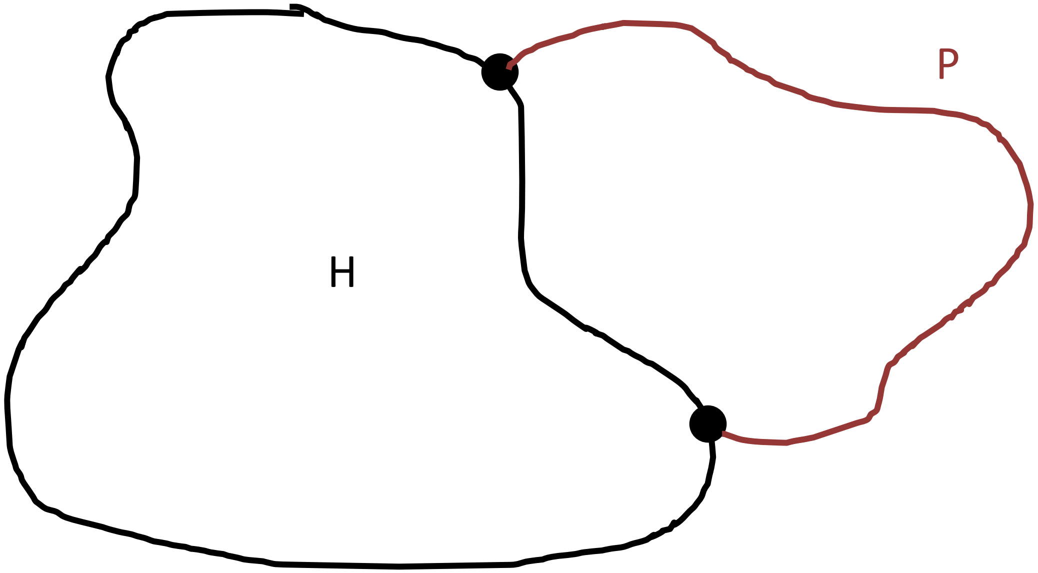


Block Tree

<u>Theorem 4.1.1:</u> The block graph is always a tree.

Ears

Definition: Given a subgraph H of G an *ear* is a path P in G starting and ending at vertices of H but with no intermediate vertices in H.

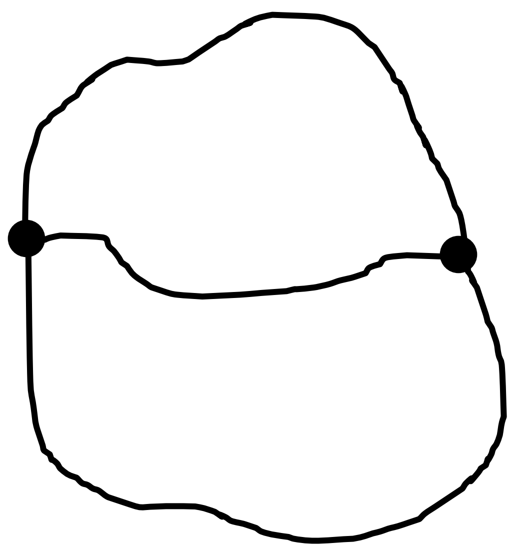


Decomposition

Theorem 4.2.1: Any block is either a K_2 or can be obtained by starting with a cycle and adding ears.

Theta Graphs

Definition: A *theta* in a graph is a pair of vertices with three vertex-disjoint paths between them.



Conclusion

Proposition 4.1.2: Any connected, theta-less graph G is a tree of cycles and K_2s.

Another Theorem on Blocks

Theorem 4.2.4: For a finite graph G with at least 3 vertices, the following are equivalent:

1) G is a single block
2) Any two edges of G are in a common cycle
3) Any two vertices of G are in a common cycle