# MODULE 9

GRAPHS AND GRAPH MODELS

PATHS AND CIRCUITS

EULER PATHS AND CIRCUITS

HAMILTON PATHS AND CIRCUITS

TREES

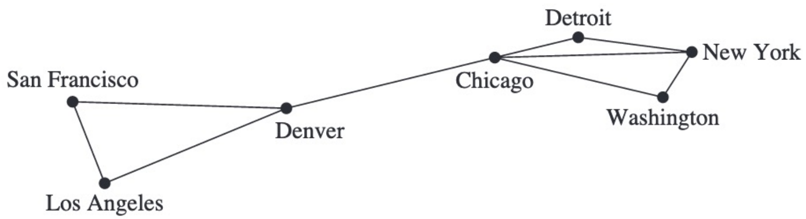
# MODULE 9

GRAPHS AND GRAPH MODELS

# DEFINITION AND EXAMPLES

A graph $G = (V, E)$ consists of a nonempty set of vertices, $V$, and a set of edges, $E$ connecting pairs of vertices.

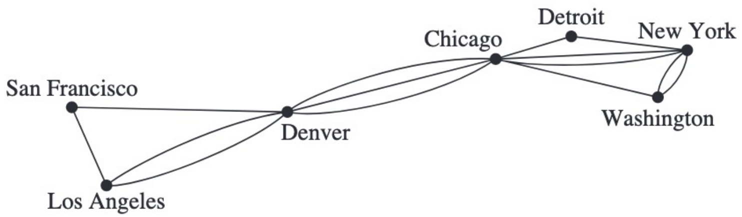
Suppose a network is made up of computer centers and communication links between computers. We can represent each location with a vertex and each communication link with an edge.




Notice that each edge connects two different vertices, i.e. no edge connects a vertex to itself. Also, no two different edges connect the same pair of vertices.

A graph in which each edge connects two different vertices, and no two edges connect the same pair of vertices is called a ***simple graph***

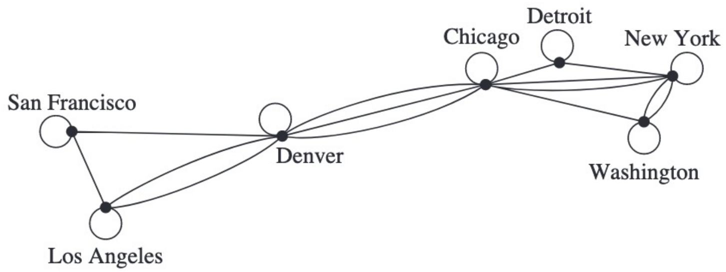
A computer network may contain multiple links between computer centers. In this case we would need more than one edge connecting two vertices.




Graphs that have multiple edges connecting the same vertices are called ***multigraphs***

# MORE EXAMPLES

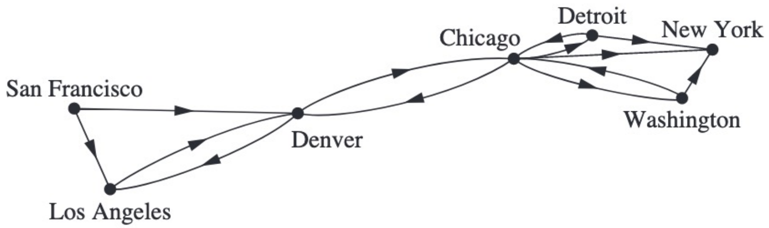
Now consider a computer network where a communication link connects a data center with itself, perhaps a feedback loop for diagnostic purposes.




An edge that connects a vertex to itself is called a ***loop***

In an undirected graph, edges are sometimes denoted $\{u, v\}$ indicating the order does not matter.

So far, the examples have been ***undirected graphs***. However, there may be the situation in our computer network example where links may only operate in one direction.



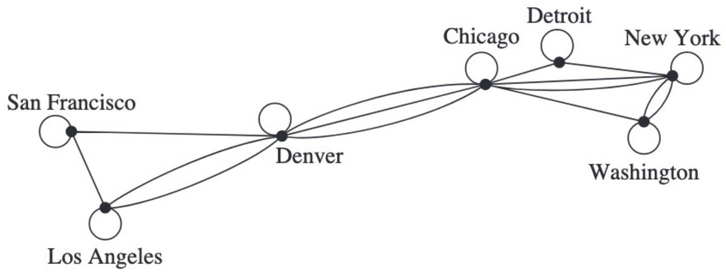

In this example we now have a ***directed graph*** that indicates the direction that the links will operate.

A ***directed graph***, sometimes called a digraph, $G = (V, E)$ consists of a nonempty set of vertices $V$ and a set of directed edges $E$.

In a directed graph the edges are sometimes denoted $(u, v)$ indicating order does matter.

# TERMINOLOGY

Two vertices $u$ and $v$ in an undirected graph $G$ are called adjacent in $G$ if $u$ and $v$ are endpoints of an edge $e$ of $G$. Such an edge $e$ is called incident with the vertices $u$ and $v$ and is said to connect $u$ and $v$.




For example, Chicago and New York are adjacent vertices.

The degree of Chicago is $9$

The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex. The degree of a vertex $v$ is typically denoted $\mathbf{deg}(v)$.

<u>Handshaking Theorem</u>

Let $G = (V, E)$ be an undirected graph with $m$ edges. Then

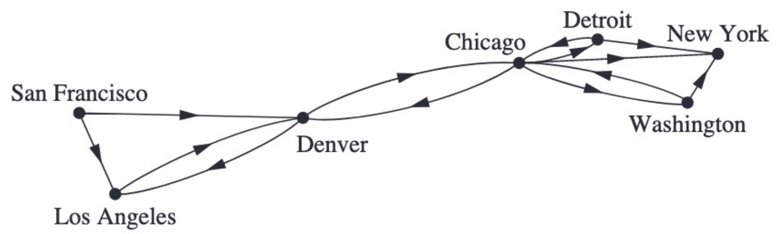
$$2m = \sum_{v \in V} \deg(v)$$

# TERMINOLOGY

When $(u, v)$ is an edge of the directed graph $G$, the vertex $u$ is called the initial vertex of $(u, v)$ and $v$ is called the terminal or end vertex of $(u, v)$. The initial vertex and terminal vertex of a loop are the same.

For the directed edge between San Francisco and Los Angeles, San Francisco is the initial vertex and Los Angeles is the terminal vertex.

There are two directed edges connecting Denver and Chicago. In one direction Chicago is the initial vertex. In the other direction Chicago is the terminal vertex.




The in-degree of Chicago is $3$ and the out-degree of Chicago is $4$

The in-degree of a vertex $v$, denoted $deg^{-}(v)$, is the number of edges with $v$ as their terminal vertex. The out-degree of $v$, denoted $deg^{+}(v)$, is the number of edges with $v$ as their initial vertex.

Theorem

Let $G = (V, E)$ be a graph with directed edges. Then

$$\sum_{v \in V} deg^{-}(v) = \sum_{v \in V} deg^{+}(v) = |E|$$

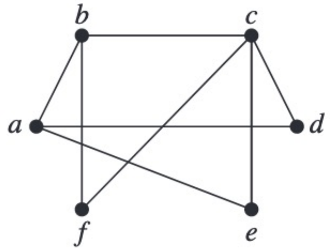
# MODULE 9

PATHS AND CIRCUITS

# PATHS AND CIRCUITS

A ***path*** is a sequence of edges that begins at a vertex and travels from vertex to vertex along the edges of a graph.

A path of length greater than zero that begins and ends at the same vertex is called a ***circuit***.

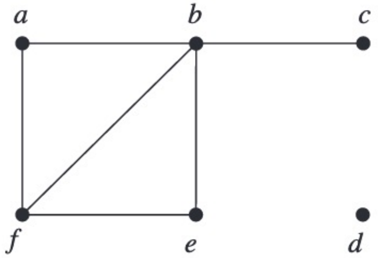



An example of a path from $c$ to $a$ would be $cda$ or $cfba$

An example of a circuit would be $cdabc$

This is a connected graph

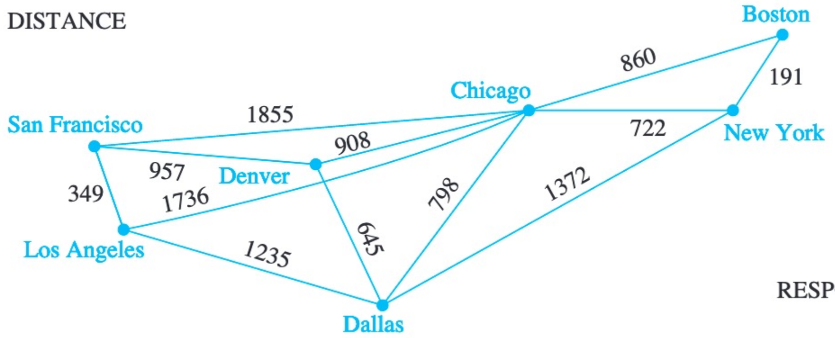
An undirected graph is connected if there is a path between every pair of distinct vertices of the graph.

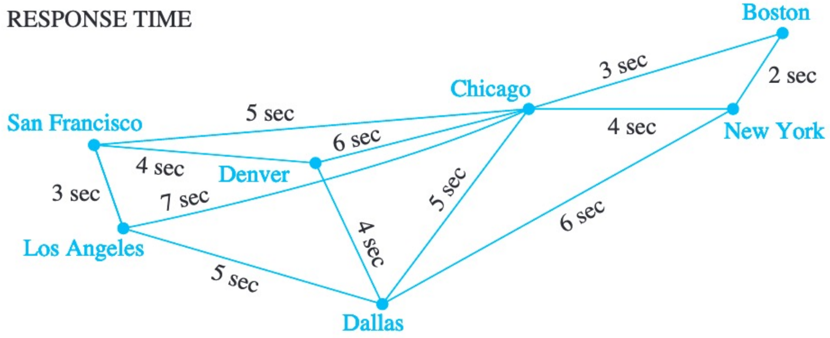


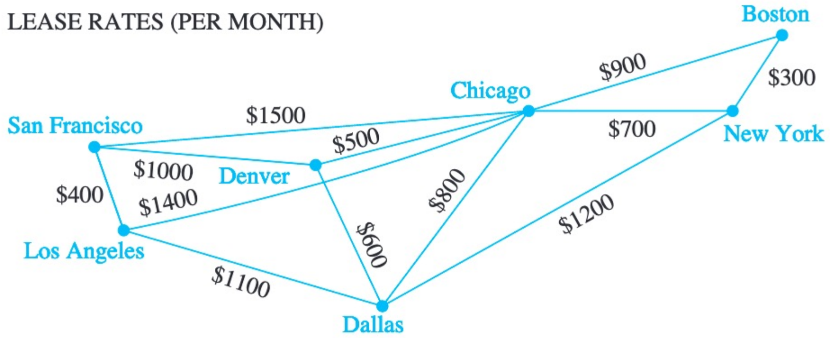

This is a disconnected graph since no path exists between $d$ and any other vertex.

# WEIGHTED GRAPHS

Occasionally, weights are assigned to the edges of a graph. For example, distances, costs, etc.



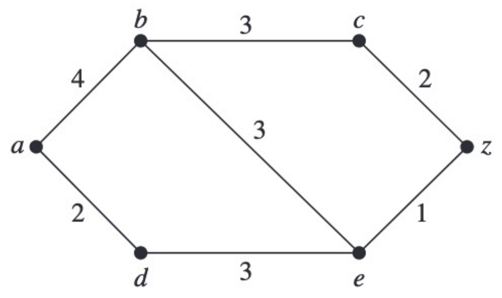



# DIJKSTRA'S ALGORITHM

What is the shortest path between $a$ and $z$?

Approach: Find the length of a shortest path from $a$ to successive vertices until $z$ is reached

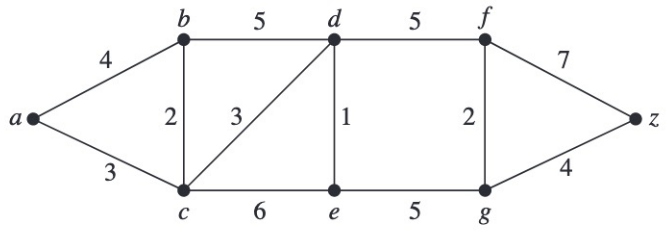



| Set of vertices | Adjacent vertices | Paths to consider | New set of vertices |
|---|---|---|---|
| $\{a\}$ | $\{b, d\}$ | $ab = 4$<br>$ad = 2$ | $\{a, d\}$ |
| $\{a, d\}$ | $\{b, e\}$ | $ab = 4$<br>$ade = 5$ | $\{a, d, b\}$ |
| $\{a, d, b\}$ | $\{c, e\}$ | $abc = 7$<br>$abe = 7$<br>$ade = 5$ | $\{a, d, b, e\}$ |
| $\{a, d, b, e\}$ | $\{c, z\}$ | $abc = 7$<br>$adez = 6$ | $\{a, d, b, e, z\}$ |

Shortest path is $adez$

# YOUR TURN

Use Dijkstra's Algorithm to find the shortest path from $a$ to $z$ in the given graph and find the length of that path




<u>Answer</u>: The shortest path is $acdegz$ with a length of $16$

# MODULE 9

EULER PATHS AND CIRCUITS

# DEFINITIONS

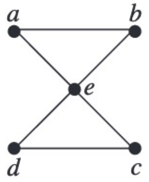
A path or circuit is called simple if it does not contain the same edge more than once

An ***Euler path*** in $G$ is a simple path containing every edge of $G$

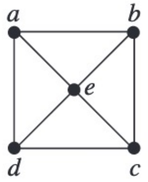
An ***Euler circuit*** in $G$ is a simple circuit containing every edge of $G$

Note: If $G$ has an Euler circuit, then $G$ necessarily has an Euler path

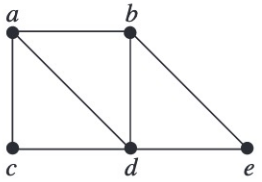
Which of these graphs have an Euler path? Which have an Euler circuit?




$G_1$




$G_2$



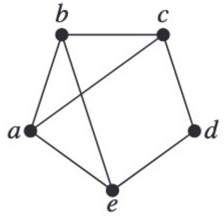

$G_3$

$G_1$ has an Euler circuit, i.e. $aecdeba$

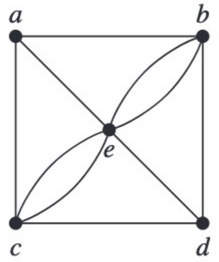
$G_2$ has no Euler path and hence no Euler circuit

$G_3$ has and Euler path, i.e. $acdebdab$, but no Euler circuit

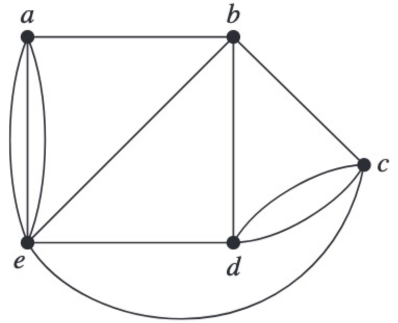
Your turn: Which of these graphs have an Euler path? Which have an Euler circuit?




$H_1$




$H_2$




$H_3$

Answer: $H_1$ has neither, $H_2$ has an Euler path but no Euler circuit, $H_3$ has an Euler circuit and hence an Euler path.

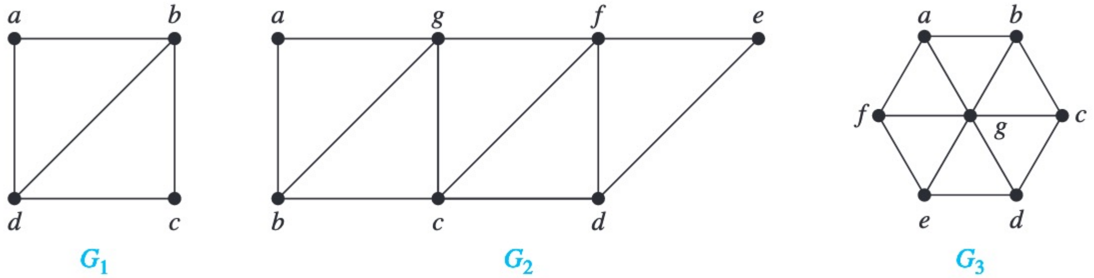
# EXISTENCE

A connected graph with at least two vertices has an Euler circuit if and only if each of its vertices has even degree

A connected graph has an Euler path but not an Euler circuit if and only if it has exactly two vertices of odd degree

Summary: An Euler path exists if exactly $0$ or $2$ vertices have odd degree. An Euler circuit exists if $0$ vertices have odd degree

Which of these graphs have an Euler path? Which have an Euler circuit?




The degrees of the vertices in $G_1$ are $2,3,2,3$. So an Euler path exists, but no Euler circuit

The degrees of the vertices in $G_2$ are $2, 3, 4, 3, 2, 4, 4$. So an Euler path exists, but no Euler circuit.

There are $6$ vertices with odd degree in $G_3$ so no Euler path exists and hence, no Euler circuit.

Your turn: Find an Euler path for $G_1$ and $G_2$. [*Hint: Any Euler path must have the two vertices of odd degree as its endpoints*]

Answers will vary, but one Euler path for $G_1$ is $dabcdb$ and one Euler path for $G_2$ is $bagfedcgbcfd$

# FLEURY'S ALGORITHM

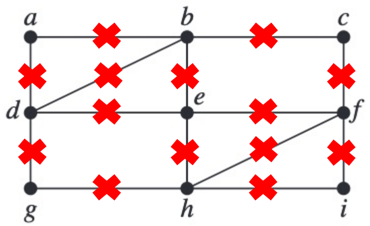
1. If finding an Euler path and the graph has two vertices of odd degree, begin at one of those vertices. If finding an Euler circuit, begin and any vertex.

2. Choose any edge leaving your current vertex, provided deleting that edge will not separate the graph into two disconnected sets of edges.

3. Add that edge to your circuit (path) and delete it from the graph.

4. Repeat 2 & 3 until a circuit (path) is found.

Find an Euler circuit

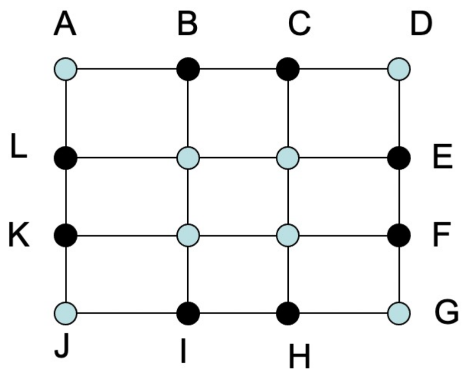


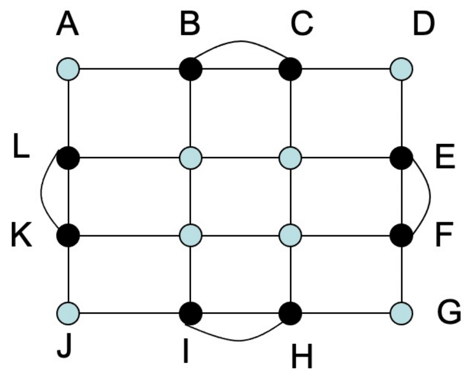

$$a \to b \to c \to f \to i \to h \to e \to f \to h \to g \to d \to e \to b \to d \to a$$

# EULERIZATION

***Eulerization*** is the process of adding edges to a graph so that an Euler circuit exists.

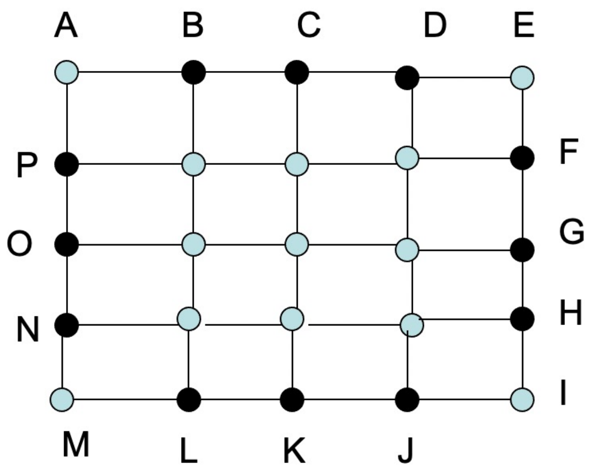
Edges are duplicated to connect vertices with odd degree. When two vertices of odd degree are not directly connected (not adjacent), duplicate edges in a path connecting the two.



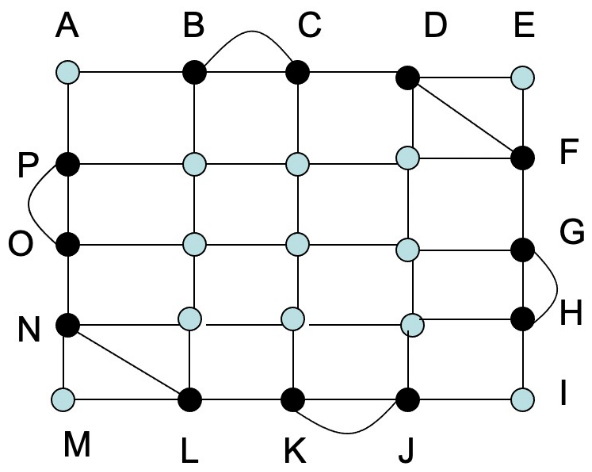

Vertices with odd degree
are shown in black

# EXAMPLE

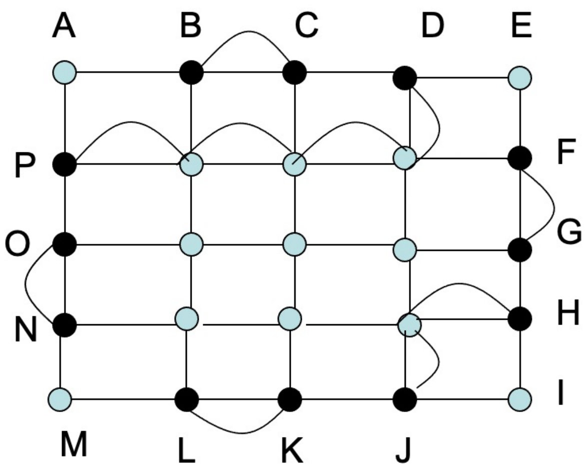



Eulerize the graph shown

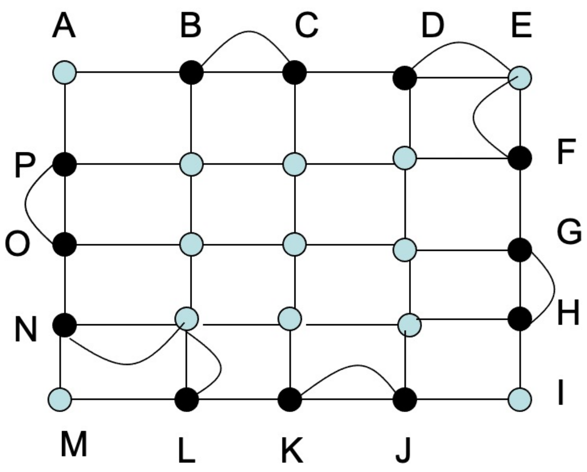
Not permitted




Inefficient




Optimal



# MODULE 9

HAMILTON PATHS AND CIRCUITS

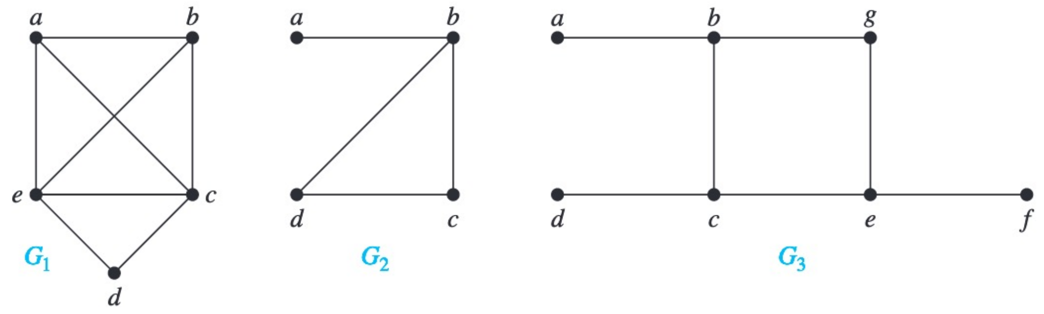
# DEFINITIONS

A ***Hamilton path*** is a simple path in a graph $G$ that passes through every vertex exactly once.

A ***Hamilton circuit*** is a simple circuit in a graph $G$ that passes through every vertex exactly once.

Note: The term "simple" implies we cannot repeat edges. However, we do not have to use every edge like Euler paths and circuits.

Which of these graphs have a Hamilton circuit or, if not, a Hamilton path?




$G_1$ has a Hamilton circuit and hence, a Hamilton path, i.e. $abcdea$

$G_2$ has no Hamilton circuit, but does have a Hamilton path, i.e. $abcd$

$G_3$ does not have a Hamilton path and hence, no Hamilton circuit

# EXISTENCE

There are no known simple necessary and sufficient criteria for the existence of Hamilton circuits.

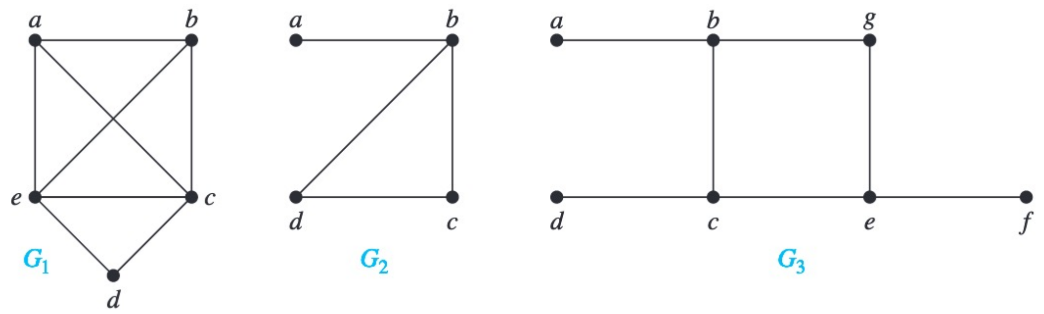
However, there are many theorems that provide sufficient conditions for the existence of a Hamilton circuit.

A graph with a vertex of degree one cannot have a Hamilton circuit, because in a Hamilton circuit, each vertex is incident with two edges in the circuit.

If $G$ is a simple graph with $n$ vertices where $n \geq 3$ and the degree of every vertex in $G$ is at least $n/2$, then $G$ has a Hamilton circuit.

If $G$ is a simple graph with $n$ vertices where $n \geq 3$ and $\deg(u) + \deg(v) \geq n$ for every pair of nonadjacent vertices $u$ and $v$ in $G$, then $G$ has a Hamilton circuit.

Which of these graphs have Hamilton paths or circuits?




$G_1$ has $5$ vertices and the sum of the degrees of each pair of nonadjacent vertices is $\geq 5$

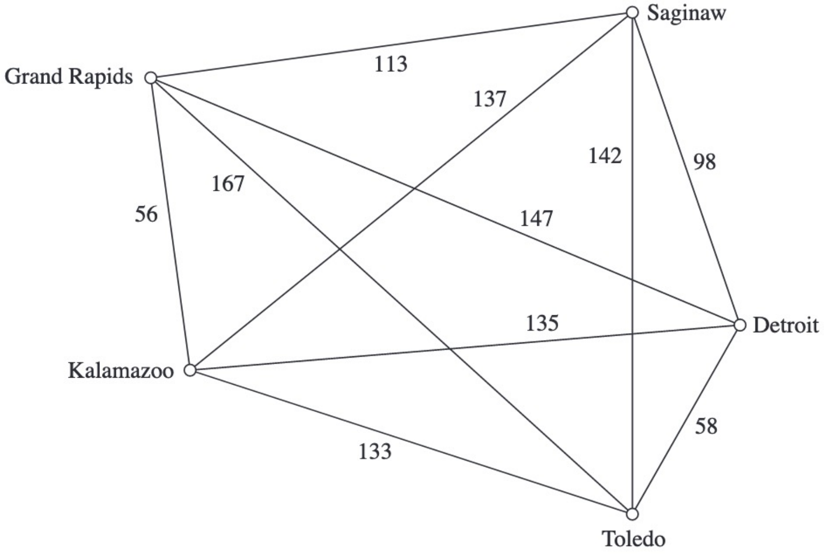
$G_2$ and $G_3$ each have at least one vertex with degree one, so no Hamilton circuit exists

# TRAVELING SALESPERSON PROBLEM

What is the shortest route a traveling salesperson should take to visit a set of cities?

This problem reduces to finding a Hamilton circuit in a complete graph, i.e. a graph that contains one edge between each pair of vertices.

This graph shows five cities with the weights of each edge given by the distance in miles between each city

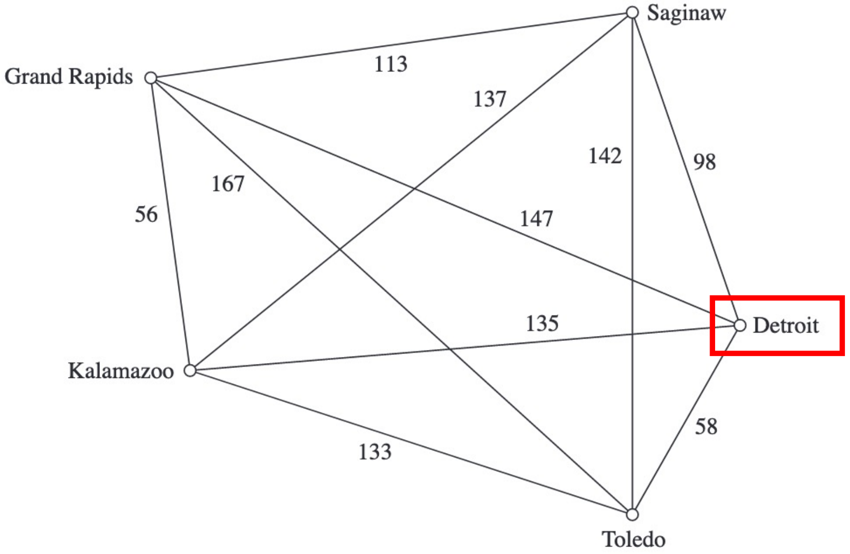



How can we find a Hamilton circuit that has the shortest length?

# BRUTE FORCE ALGORITHM

The *brute force algorithm* lists all Hamilton circuits and selects the one with the minimum total length.

This will always produce an optimum result, but is inefficient in most cases



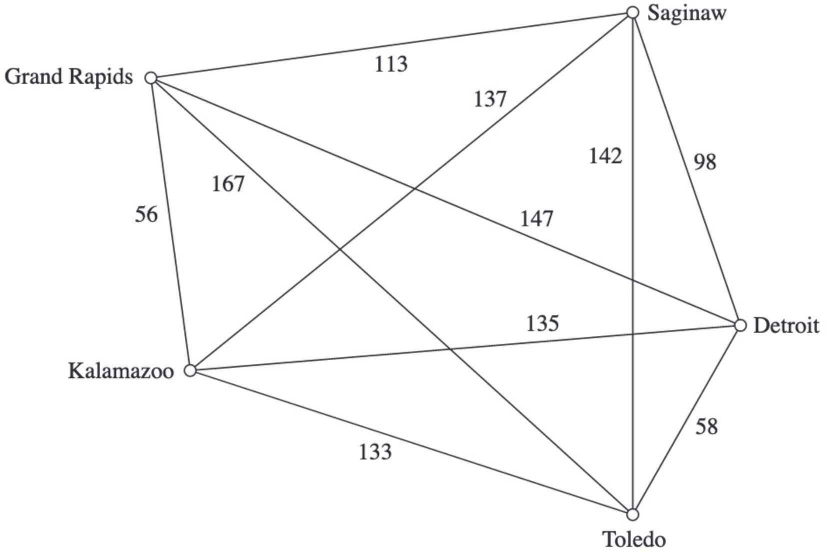

| *Route* | *Total Distance (miles)* |
|---|---|
| Detroit–Toledo–Grand Rapids–Saginaw–Kalamazoo–Detroit | 610 |
| Detroit–Toledo–Grand Rapids–Kalamazoo–Saginaw–Detroit | 516 |
| Detroit–Toledo–Kalamazoo–Saginaw–Grand Rapids–Detroit | 588 |
| Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit | 458 |
| Detroit–Toledo–Saginaw–Kalamazoo–Grand Rapids–Detroit | 540 |
| Detroit–Toledo–Saginaw–Grand Rapids–Kalamazoo–Detroit | 504 |
| Detroit–Saginaw–Toledo–Grand Rapids–Kalamazoo–Detroit | 598 |
| Detroit–Saginaw–Toledo–Kalamazoo–Grand Rapids–Detroit | 576 |
| Detroit–Saginaw–Kalamazoo–Toledo–Grand Rapids–Detroit | 682 |
| Detroit–Saginaw–Grand Rapids–Toledo–Kalamazoo–Detroit | 646 |
| Detroit–Grand Rapids–Saginaw–Toledo–Kalamazoo–Detroit | 670 |
| Detroit–Grand Rapids–Toledo–Saginaw–Kalamazoo–Detroit | 728 |

In a graph with $n$ vertices, once a starting point has been selected, there will be $(n-1)!$ different Hamilton circuits to examine

Half of those are duplicates (in reverse order), so we only need to examine $(n-1)!/2$ to find the answer but this can still be quite a large number

# NEAREST NEIGHBOR ALGORITHM

For the *nearest neighbor algorithm* (NNA), we select a starting point and move to the nearest unvisited neighbor. Repeat this until the circuit is complete.




Using the NNA:

1. The closest neighbor to Detroit is Toledo
2. The closest unvisited neighbor to Toledo is Kalamazoo
3. The closest unvisited neighbor to Kalamazoo is Grand Rapids
4. The closest unvisited neighbor to Grand Rapids is Saginaw
5. Return to Detroit

With the *repeated nearest neighbor algorithm*, we work the the NNA at each vertex then choose the optimum circuit

Recall, the shortest circuit starting with Detroit was:

Detroit–Toledo–Kalamazoo–Grand Rapids–Saginaw–Detroit

# MODULE 9

TREES

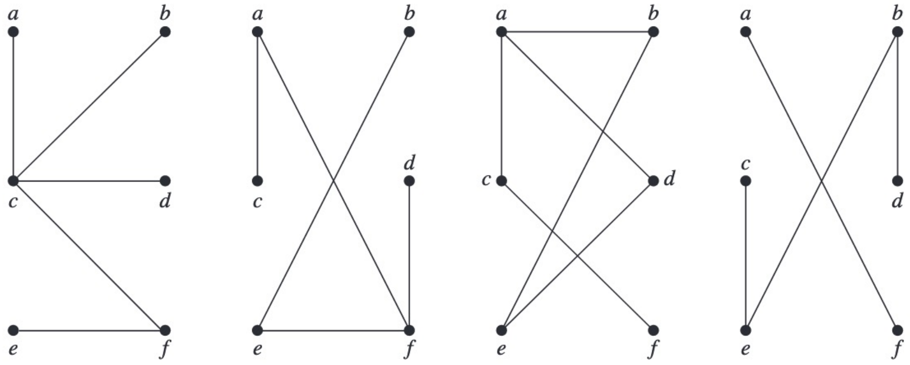
# TREES

A ***tree*** is a connected undirected graph with no simple circuits

Recall: A simple circuit does not contain the same edge more than once

Note: This implies a tree cannot contain multiple edges or loops. So any tree must be a simple graph

Which of these graphs are trees?



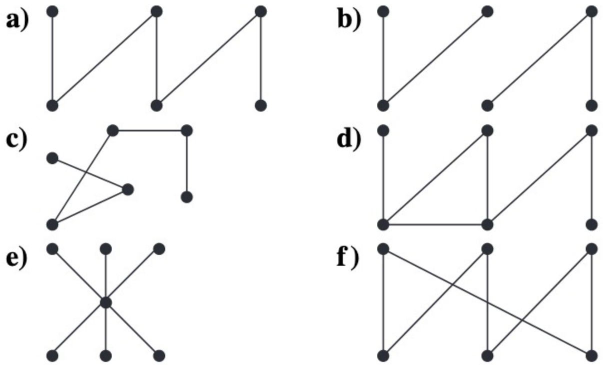

$G_1$ $G_2$ $G_3$ $G_4$

$G_1$ and $G_2$ are trees because both are connected with no simple circuits

$G_3$ is not a tree because $ebade$ is a simple circuit

$G_4$ is not a tree because it is not connected

Your turn: Which of these graphs are trees?

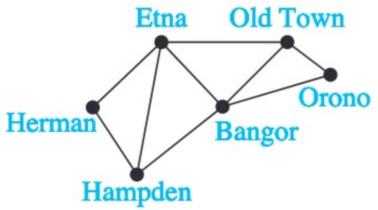



Answer: $a, c,$ and $e$

# SPANNING TREES

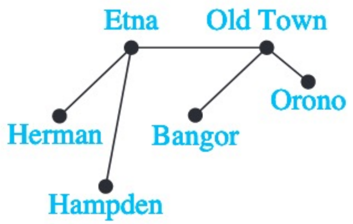
A ***spanning tree*** of a graph $G$ is a tree containing every vertex of $G$

Road system in Maine



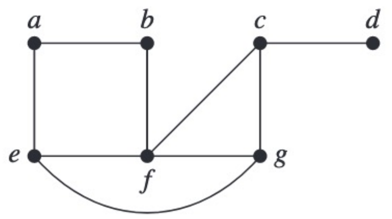

Highway department wants to plow the fewest roads so that there is always a cleared road between any two towns

Spanning tree of roads to plow

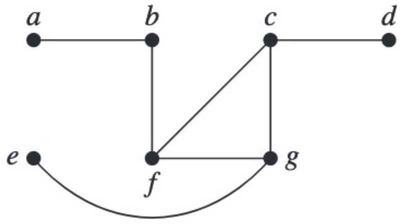


# EXAMPLE

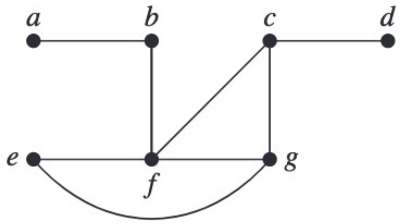
Find a spanning tree of the graph shown



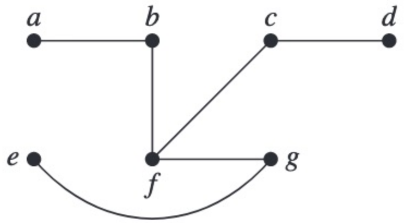

This graph is not a tree because it contains simple circuits




Remove edge $\{e, f\}$ to eliminate another simple circuit

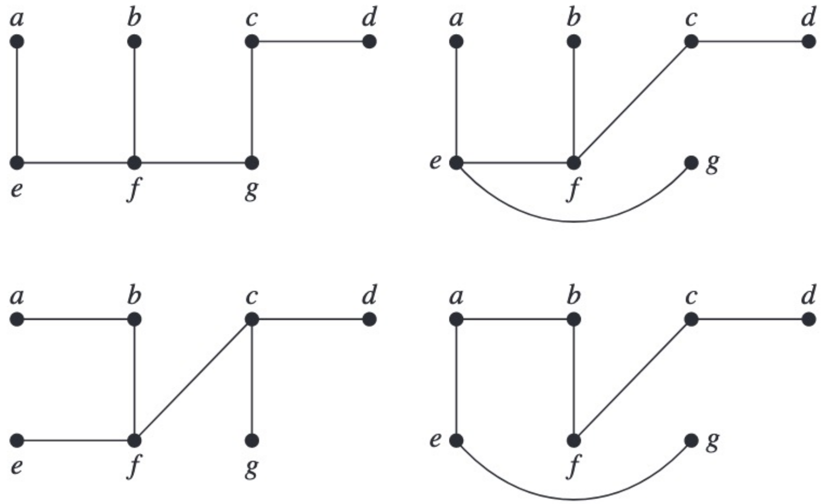



Remove edge $\{a, e\}$ to eliminate one of the simple circuits




Remove edge $\{c, g\}$ to produce a simple graph with no simple circuits

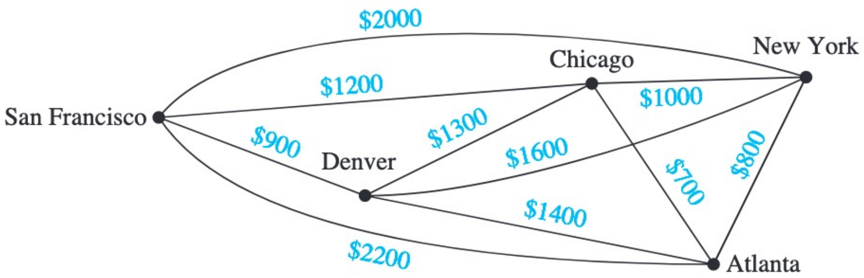
Other spanning trees of the original graph

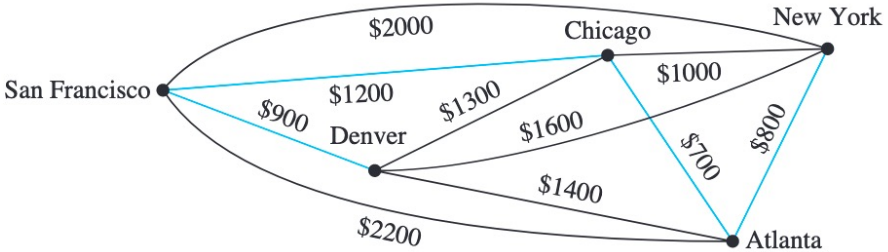


# MINIMUM COST SPANNING TREES

A minimum (cost) spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges

Weighted graph showing monthly lease costs for lines in a computer network

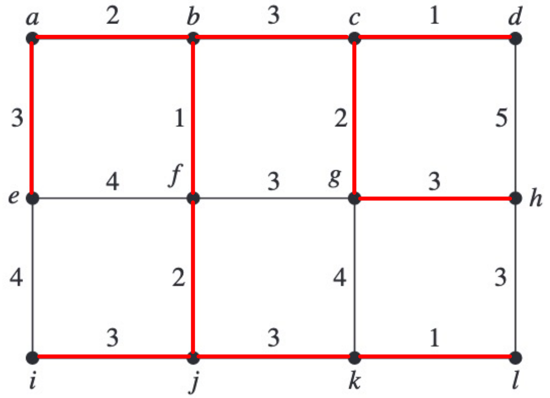



| Choice | Edge | Cost |
|---|---|---|
| 1 | {Chicago, Atlanta} | $ 700 |
| 2 | {Atlanta, New York} | $ 800 |
| 3 | {Chicago, San Francisco} | $1200 |
| 4 | {San Francisco, Denver} | $ 900 |
| | Total: | $3600 |

# KRUSKAL'S ALGORITHM

1. Order the edges in ascending order based on their weights
2. Repeatedly select unused edges provided the edge will not create a circuit
3. Repeat until a spanning tree is formed



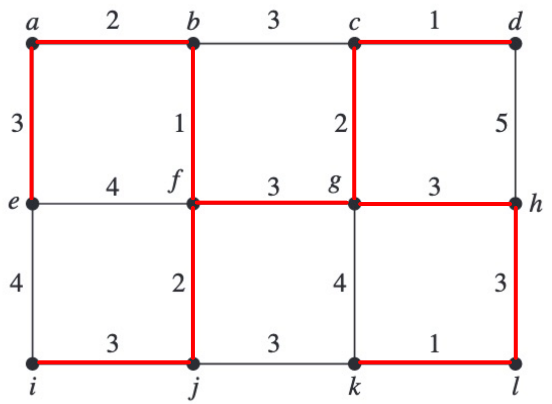

| | | |
|---|---|---|
| $\{c,d\}$ | 1 | |
| $\{k,l\}$ | 1 | |
| $\{b,f\}$ | 1 | |
| $\{c,g\}$ | 2 | |
| $\{a,b\}$ | 2 | |
| $\{f,j\}$ | 2 | |
| $\{b,c\}$ | 3 | |
| $\{f,g\}$ | 3 | Adding $\{f,g\}$ creates a circuit |
| $\{j,k\}$ | 3 | |
| $\{g,h\}$ | 3 | |
| $\{i,j\}$ | 3 | |
| $\{a,e\}$ | 3 | All vertices have been used |
| $\{h,l\}$ | 3 | |
| $\{e,f\}$ | 4 | |
| $\{e,i\}$ | 4 | |
| $\{g,k\}$ | 4 | |
| $\{d,h\}$ | 5 | |

| | |
|---|---|
| $\{c,d\}$ | 1 |
| $\{k,l\}$ | 1 |
| $\{b,f\}$ | 1 |
| $\{c,g\}$ | 2 |
| $\{a,b\}$ | 2 |
| $\{f,j\}$ | 2 |
| $\{b,c\}$ | 3 |
| $\{j,k\}$ | 3 |
| $\{g,h\}$ | 3 |
| $\{i,j\}$ | 3 |
| $\{a,e\}$ | 3 |

total $= 24$

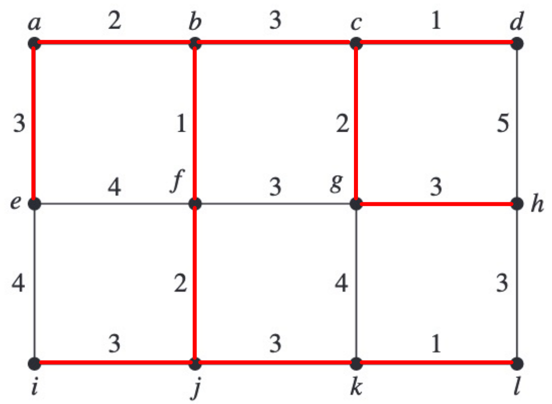
# PRIM'S ALGORITHM

1. Select an edge with minimal weight
2. Select an edge of minimum weight incident to a vertex in the tree that does not create a simple circuit.
3. Repeat step 2 until a spanning tree is found




| | |
|---|---|
| $\{b, f\}$ | 1 |
| $\{a, b\}$ | 2 |
| $\{f, j\}$ | 2 |
| $\{a, e\}$ | 3 |
| $\{i, j\}$ | 3 |
| $\{f, g\}$ | 3 |
| $\{c, g\}$ | 2 |
| $\{c, d\}$ | 1 |
| $\{g, h\}$ | 3 |
| $\{h, l\}$ | 3 |
| $\{k, l\}$ | 1 |

total = 24




Compare to Kruskal's algorithm

| | |
|---|---|
| $\{c, d\}$ | 1 |
| $\{k, l\}$ | 1 |
| $\{b, f\}$ | 1 |
| $\{c, g\}$ | 2 |
| $\{a, b\}$ | 2 |
| $\{f, j\}$ | 2 |
| $\{b, c\}$ | 3 |
| $\{j, k\}$ | 3 |
| $\{g, h\}$ | 3 |
| $\{i, j\}$ | 3 |
| $\{a, e\}$ | 3 |

total = 24

# QUESTIONS?