Chapter 3: Graph Theory

Definition (Graphs, Vertices, and Edges)

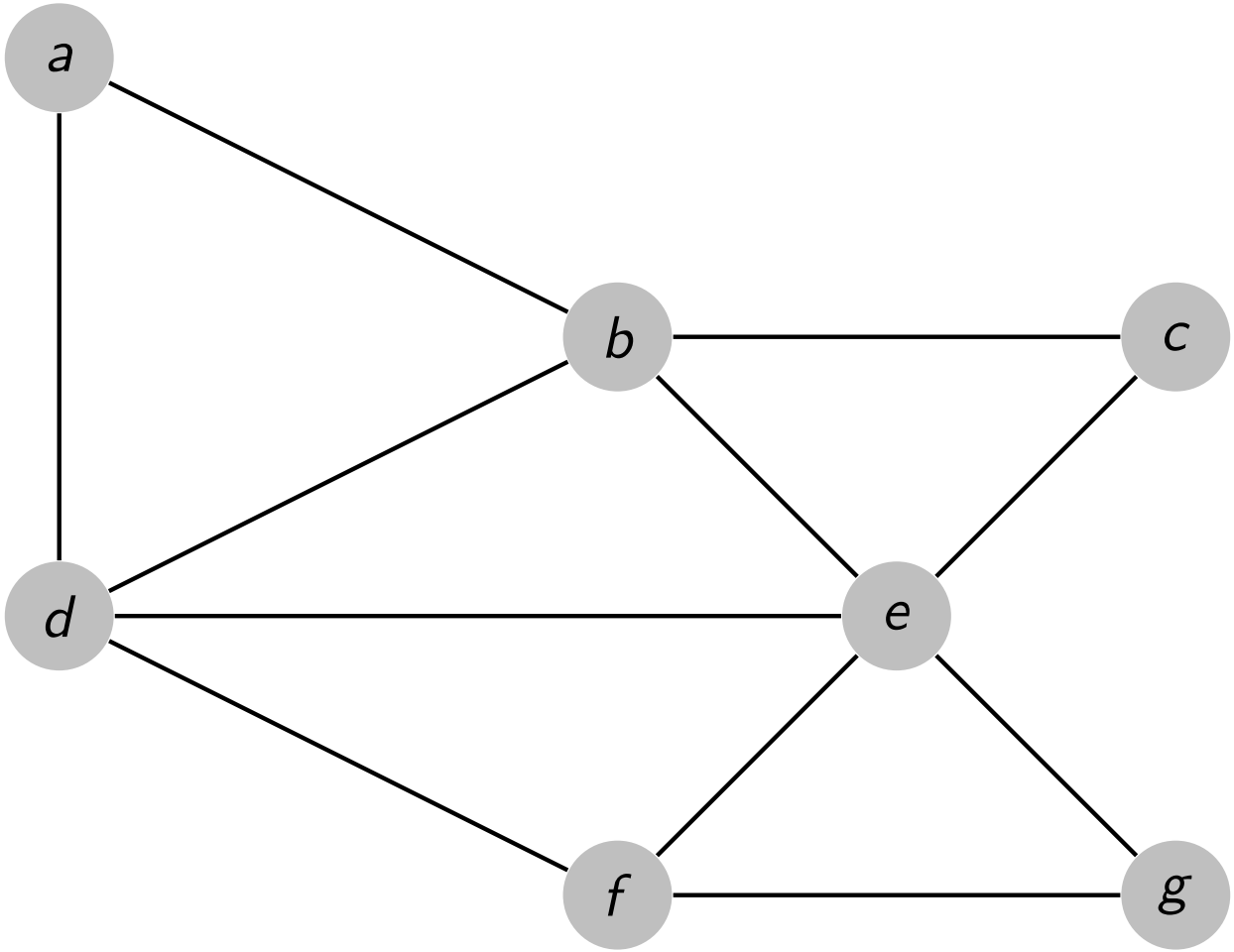
A **graph** consists of a set of dots, called **vertices**, and a set of **edges** connecting pairs of vertices.

Definition (Vertex)

A **vertex** is a dot in the graph that could represent an intersection of streets, a land mass, or a general location, like “work‘ or “school“. Vertices are often connected by edges. Note that vertices only occur when a dot is explicitly placed, not whenever two edges cross. Imagine a freeway overpass –the freeway and side street cross, but it is not possible to change from the side street to the freeway at that point, so there is no intersection and no vertex would be placed.

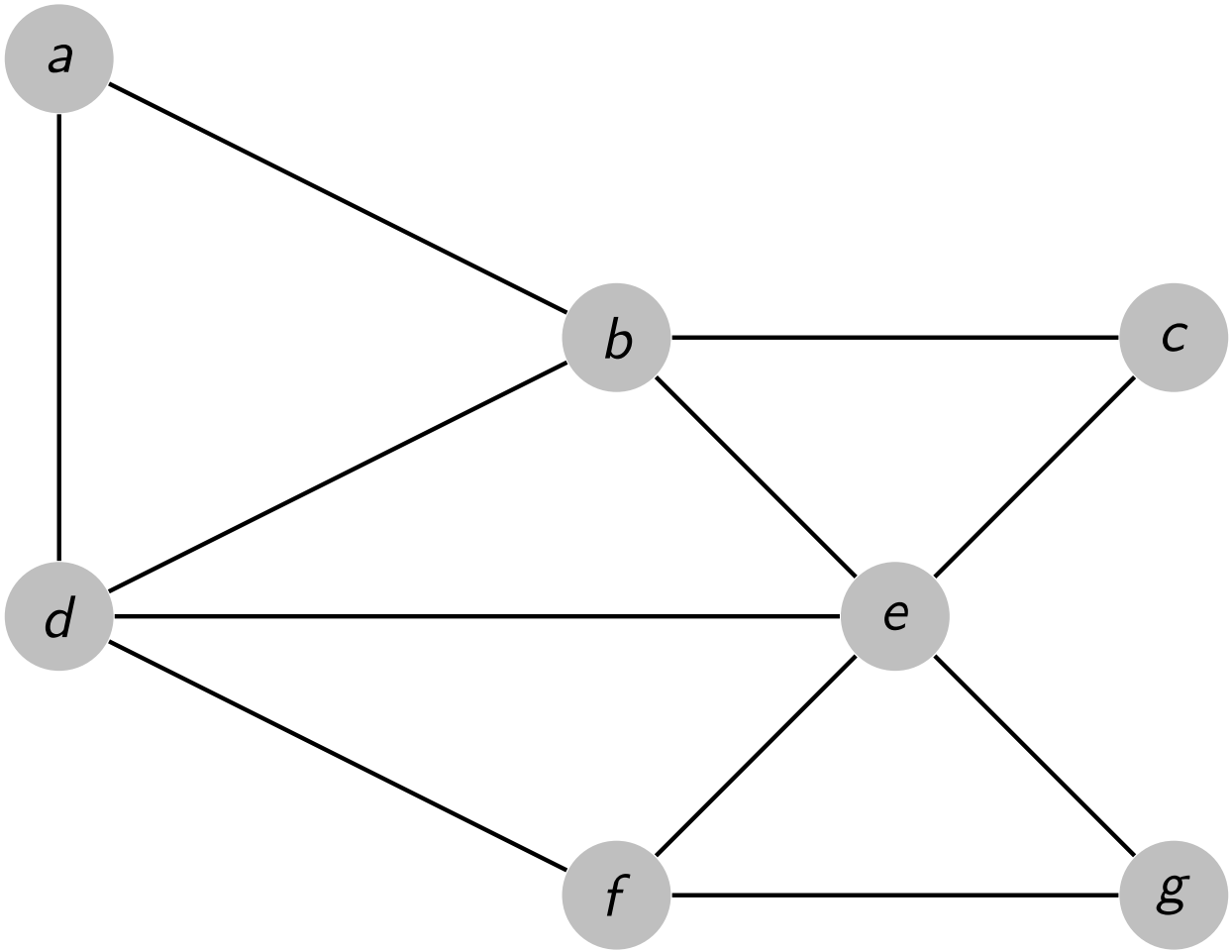
Definition (Edges)

Edges connect pairs of vertices. An edge can represent a physical connection between locations, like a street, or simply that a route connecting the two locations exists, like an airline flight.


a
b
c
d
e
f
g

Definition (Degree of a vertex)

The **degree** of a vertex is the number of edges meeting at that vertex. It is possible for a vertex to have a degree of zero or larger.

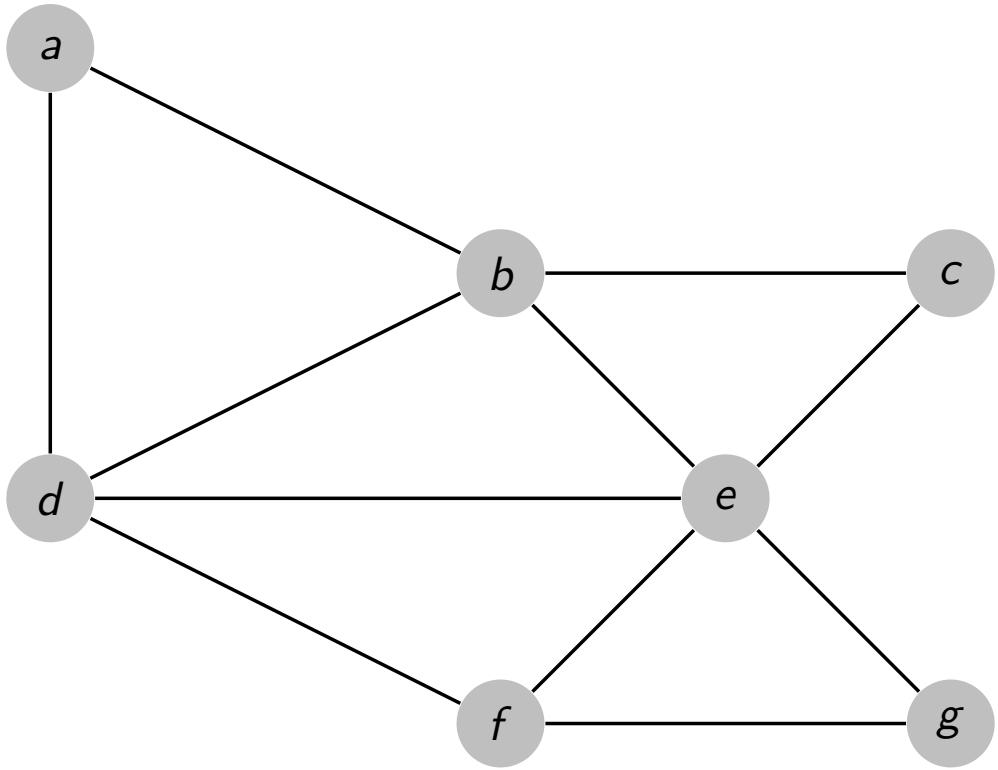


Definition (Path)

A **path** is a sequence of vertices connected by using the edges.

Definition (Circuit)

A **circuit** is a path that begins and ends at the same vertex.

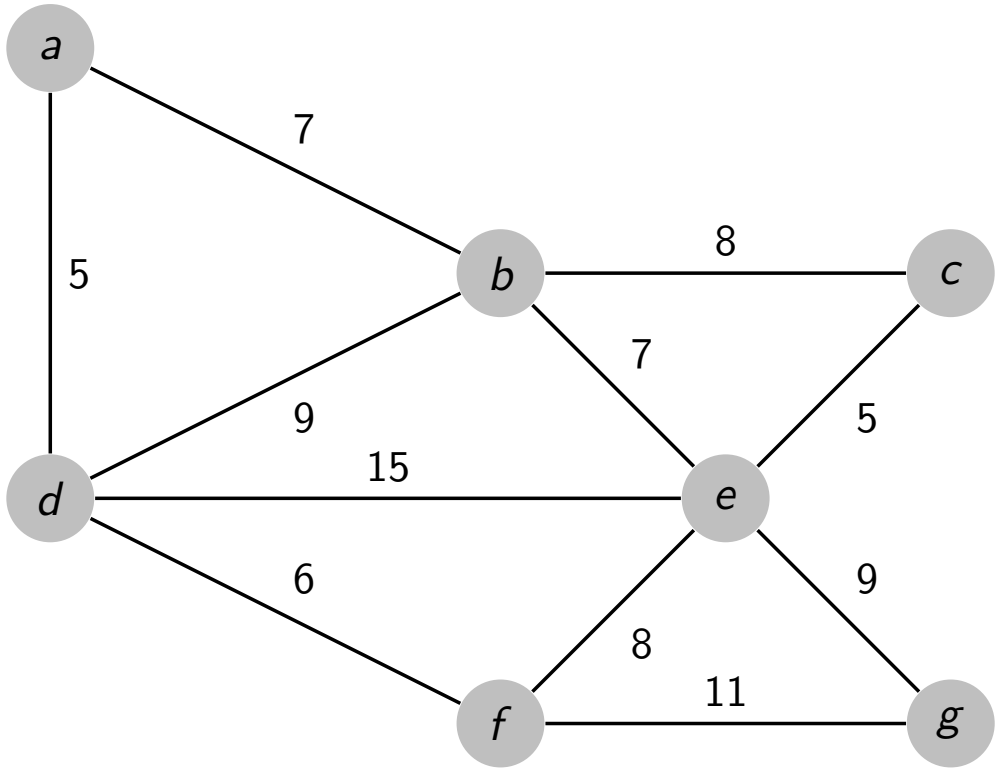


Definition (Path)

A **path** is a sequence of vertices connected by using the edges.

Definition (Circuit)

A **circuit** is a path that begins and ends at the same vertex.



Definition (Weights)

Weights are numerical values are assigned to the edges of a graph. The weights could represent the distance between two locations, the travel time, or the travel cost. It is important to note that the distance between vertices in a graph does not necessarily correspond to the weight of an edge.

Definition (Connected)

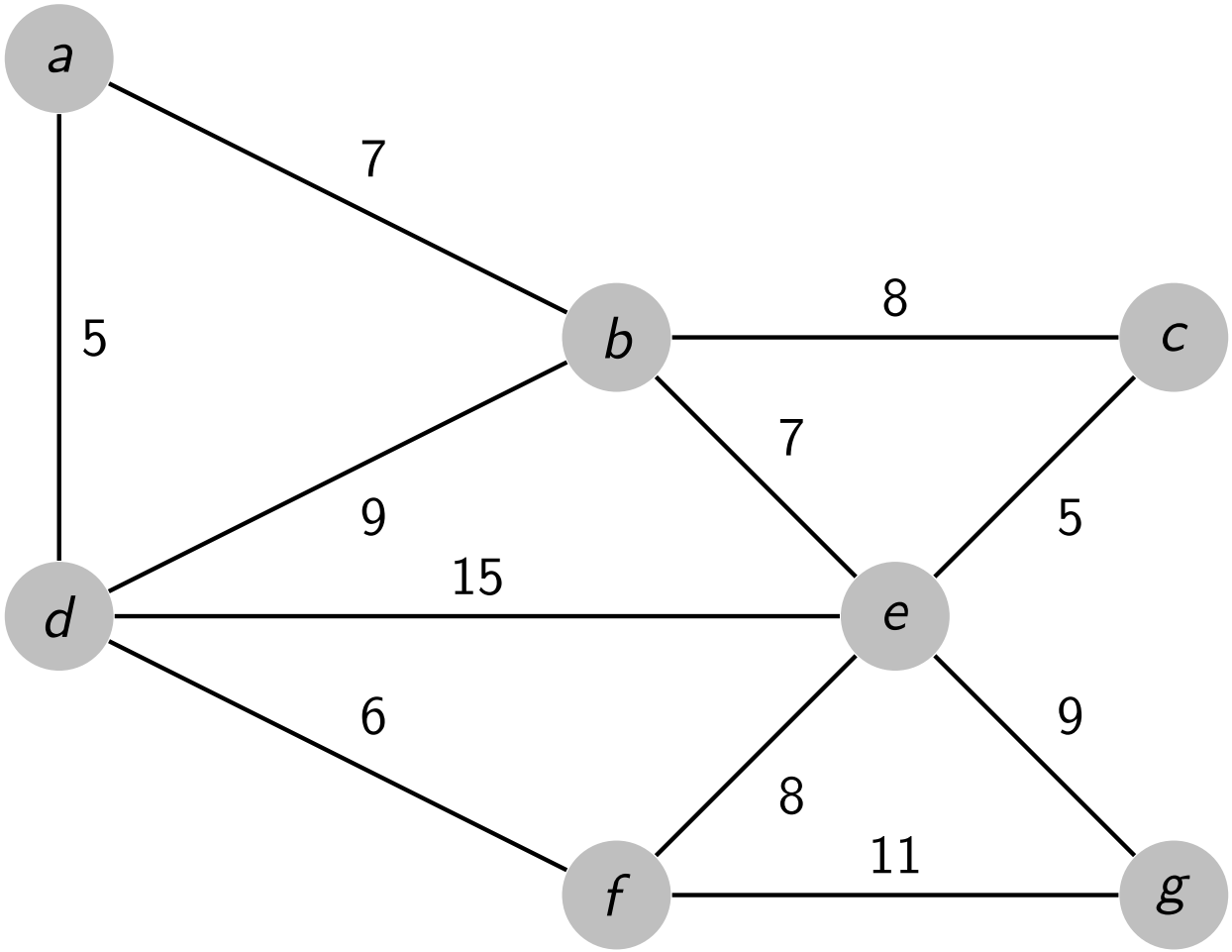
A graph is **connected** if there is a path from any vertex to any other vertex. A graph is **disconnected** if there is a pair of vertices with no path between them.

Dijkstra's Algorithm

Algorithm (Dijkstra's Algorithm)

1. Mark the ending vertex with a distance of zero. Designate this vertex as current.
2. Find all vertices leading to the current vertex. Calculate their distances to the end. Since we already know the distance the current vertex is from the end, this will just require adding the most recent edge. Don't record this distance if it is longer than a previously recorded distance.
3. Mark the current vertex as visited. We will never look at this vertex again.
4. Mark the vertex with the smallest distance as current, and re peat from step 2.

Find the shortest path from a to g.



Adjacency matrix

Dijkstra's Algorithm on adjacency matrix

	Charlotte [52]	Atlanta [19]	Nashville [25]	Knoxville [37]	**Columbia** [40]	Ralei [0]
Charlotte	*			15	14	
Atlanta		*		18	24	19
Nashville			*	18	15	25
Knoxville	15	18	18	*	14	
Columbia	14	24	15	14	8	
Raleigh		19	25			*

Definition (Loop)

A **loop** is a special type of edge that connects a vertex to itself.

Euler path and circuit

Definition (Euler Path)

1. An **Euler path** is a path that uses every edge in a graph with no repeats. Being a path, it does not have to return to the starting vertex.
2. An **Euler circuit** is a circuit that uses every edge in a graph with no repeats. Being a circuit, it must start and end at the same vertex.

Euler path and circuit

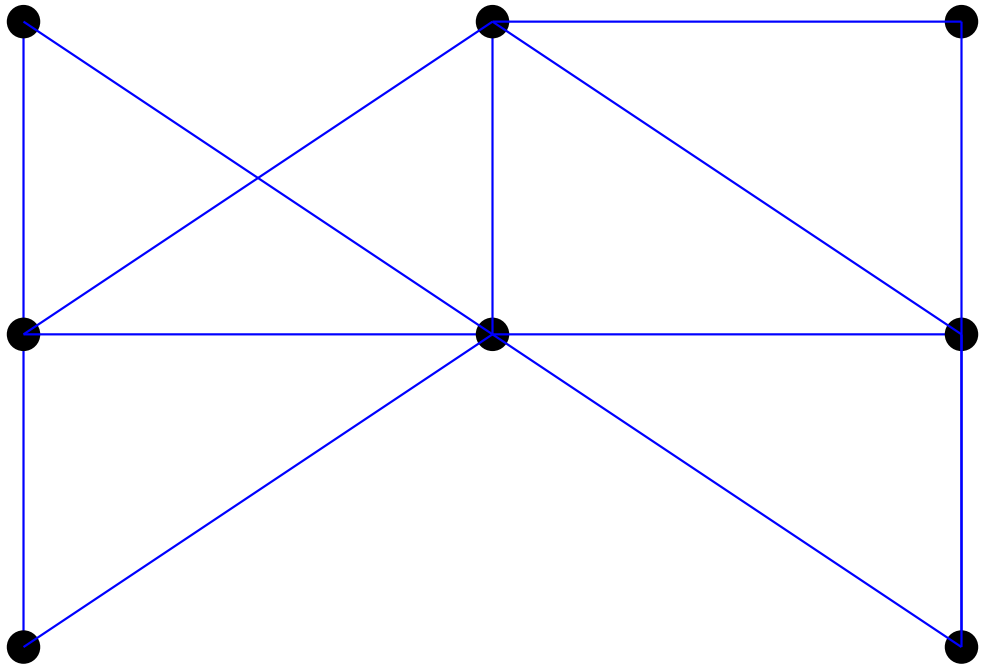
Definition (Euler Path)

1. An **Euler path** is a path that uses every edge in a graph with no repeats. Being a path, it does not have to return to the starting vertex.
2. An **Euler circuit** is a circuit that uses every edge in a graph with no repeats. Being a circuit, it must start and end at the same vertex.

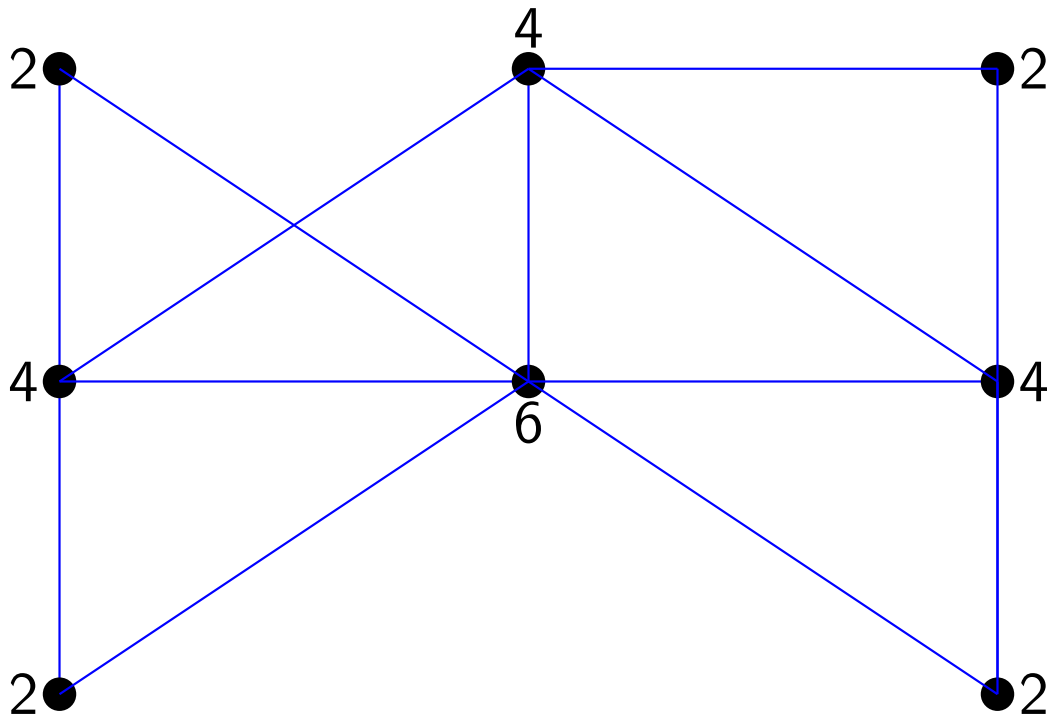
Theorem (Euler's Path and Circuit Theorems)

- *A graph will contain an Euler path if it contains at most two vertices of odd degree.*
- *A graph will contain an Euler circuit if all vertices have even degree*

Euler example



Euler example

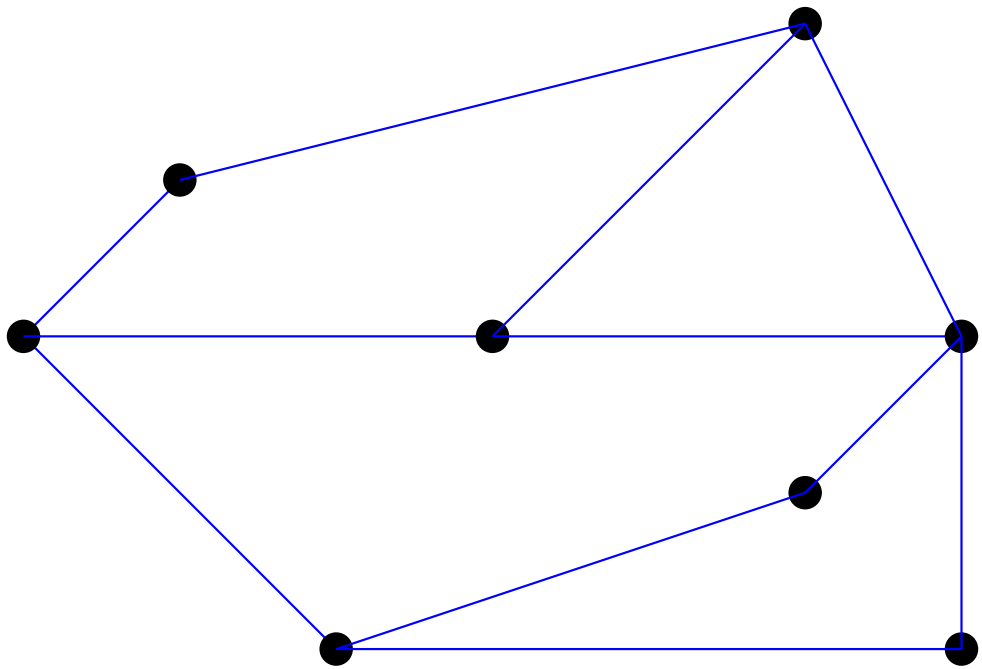


Eulerization

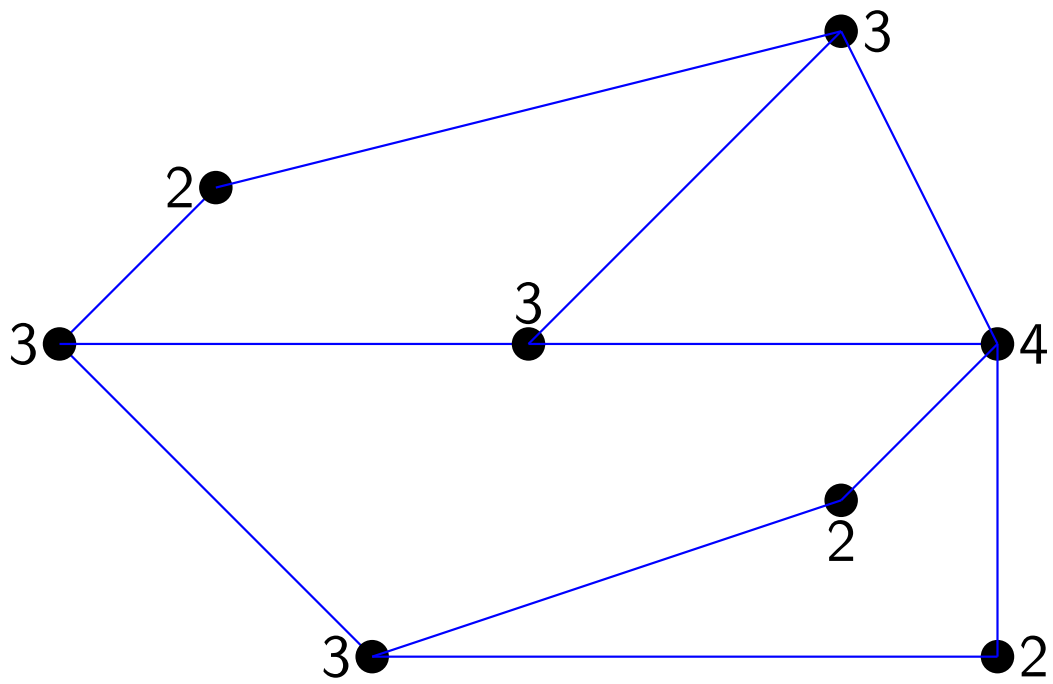
Definition (Eulerization)

Eulerization is the process of adding edges to a graph to create an Euler circuit on a graph. To eulerize a graph, edges are duplicated to connect pairs of vertices with odd degree. Connecting two odd degree vertices increases the degree of each, giving them both even degree. When two odd degree vertices are not directly connected, we can duplicate all edges in a path connecting the two.

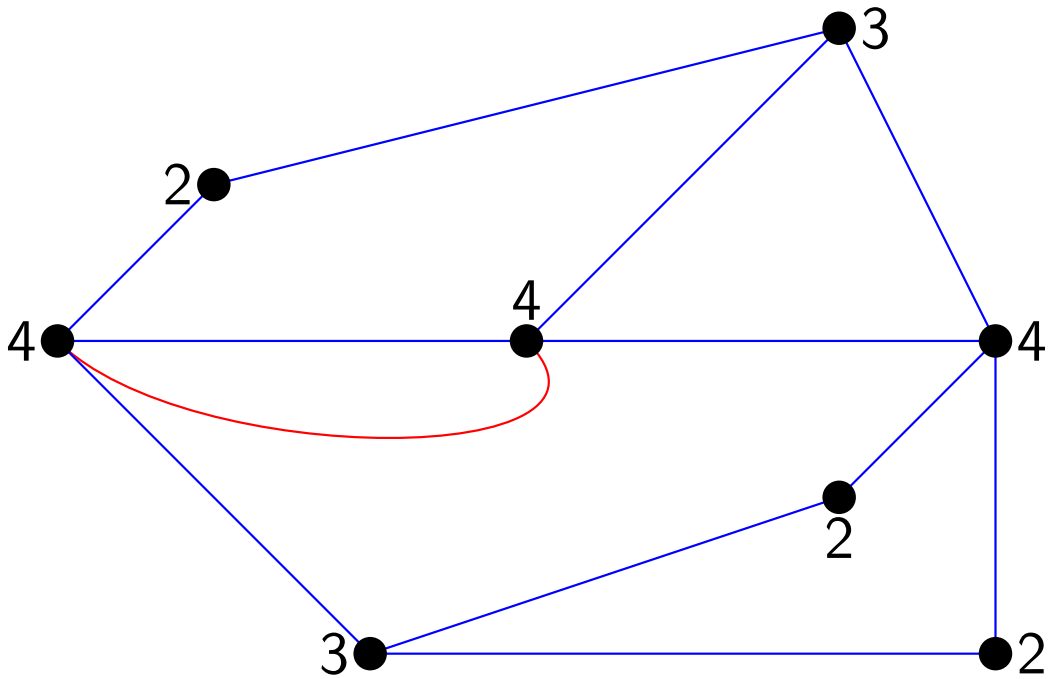
Chinese Postman Problem



Chinese Postman Problem



Chinese Postman Problem



Hamilton paths and circuits

Definition (Complete Graph)

A **complete graph** on n vertices contains exactly one edge between every pair of vertices.

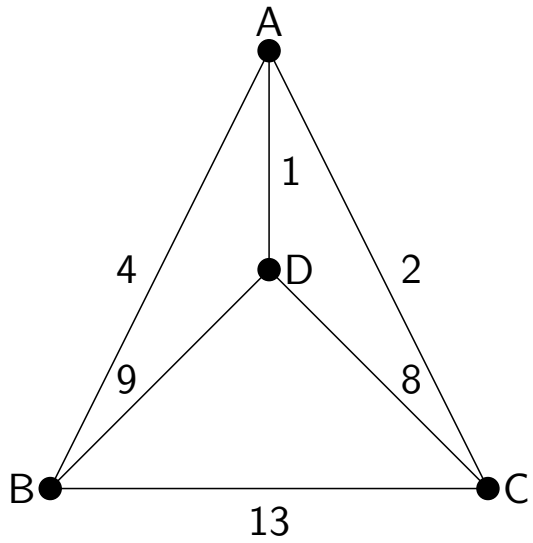
Hamilton paths and circuits

Definition (Complete Graph)

A **complete graph** on n vertices contains exactly one edge between every pair of vertices.

Definition (Hamiltonian Circuits and Paths)

A Hamiltonian circuit is a circuit that visits every vertex once with no repeats. A Hamiltonian path also visits every vertex once with no repeats, but does not have to start and end at the same vertex.

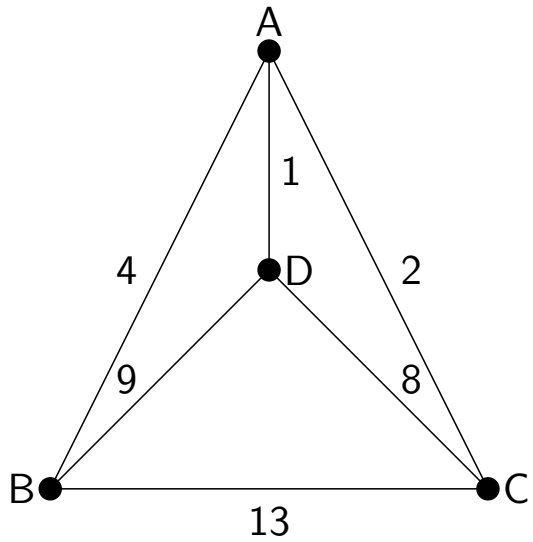


Nearest Neighbor Algorithm (NNA)

1. Select a starting point.
2. Move to the nearest unvisited vertex (the edge with smallest weight).
3. Repeat until the circuit is complete.

Nearest Neighbor Algorithm (NNA)

1. Select a starting point.
2. Move to the nearest unvisited vertex (the edge with smallest weight).
3. Repeat until the circuit is complete.

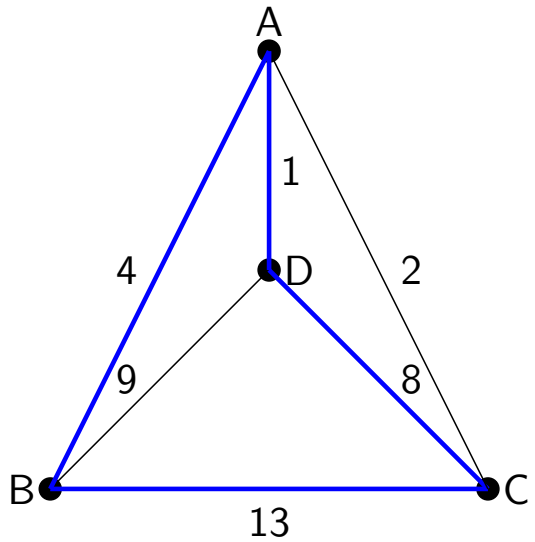


Nearest Neighbor Algorithm (NNA)

1. Select a starting point.
2. Move to the nearest unvisited vertex (the edge with smallest weight).
3. Repeat until the circuit is complete.

Nearest Neighbor Algorithm (NNA)

1. Select a starting point.
2. Move to the nearest unvisited vertex (the edge with smallest weight).
3. Repeat until the circuit is complete.



$A \to D \to C \to B \to A$, 26

Repeated Nearest Neighbor Algorithm (RNNA)

1. Do the Nearest Neighbor Algorithm starting at each vertex.
2. Choose the circuit produced with minimal total weight.

Repeated Nearest Neighbor Algorithm (RNNA)

1. Do the Nearest Neighbor Algorithm starting at each vertex.
2. Choose the circuit produced with minimal total weight.

$$B \to A \to D \to C \to B, 26$$
$$C \to A \to D \to B \to C, 25$$
$$D \to A \to C \to B \to D, 25$$

Repeated Nearest Neighbor Algorithm (RNNA)

1. Do the Nearest Neighbor Algorithm starting at each vertex.
2. Choose the circuit produced with minimal total weight.

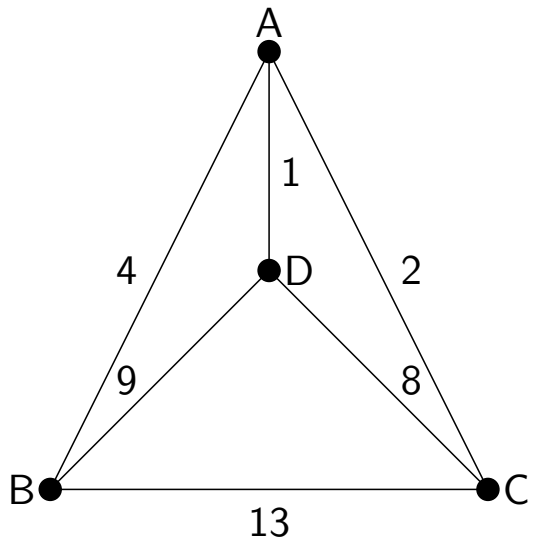
$$B \to A \to D \to C \to B,\ 26$$
$$C \to A \to D \to B \to C,\ 25$$
$$D \to A \to C \to B \to D,\ 25$$

Sorted Edges Algorithm

1. Select the cheapest unused edge in the graph.
2. Repeat step 1, adding the cheapest unused edge to the circuit, unless:
    1. adding the edge would create a circuit that doesn't contain all vertices, or
    2. adding the edge would give a vertex degree 3.
3. Repeat until a circuit containing all vertices is found.

Sorted Edges Algorithm

1. Select the cheapest unused edge in the graph.
2. Repeat step 1, adding the cheapest unused edge to the circuit, unless:
    1. adding the edge would create a circuit that doesn't contain all vertices, or
    2. adding the edge would give a vertex degree 3.
3. Repeat until a circuit containing all vertices is found.



Spanning Tree

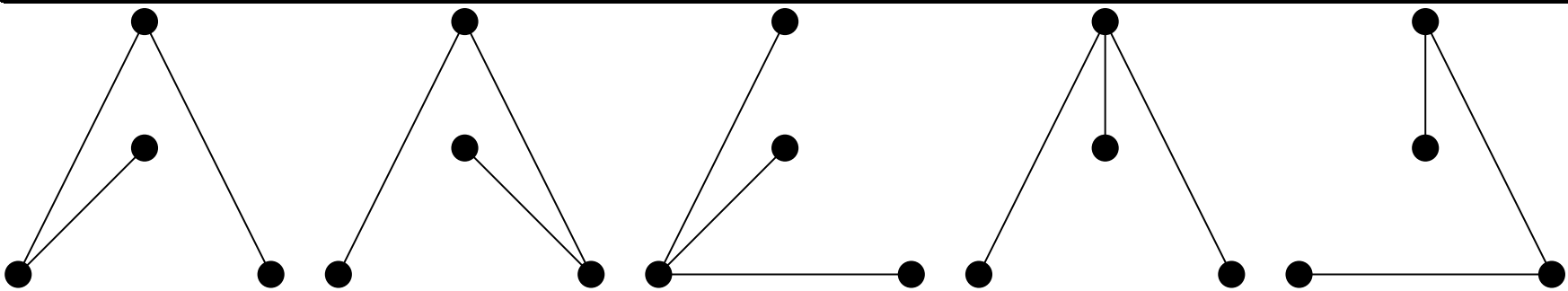
Definition (Spanning Tree)

A spanning tree is a connected graph using all vertices in which there are no circuits. In other words, there is a path from any vertex to any other vertex, but no circuits.

Spanning Tree

Definition (Spanning Tree)

A spanning tree is a connected graph using all vertices in which there are no circuits. In other words, there is a path from any vertex to any other vertex, but no circuits.



Kruskal's Algorithm

Definition (Minimum Cost Spanning Tree (MCST))

The minimum cost spanning tree is the spanning tree with the smallest total edge weight.

Kruskal's Algorithm

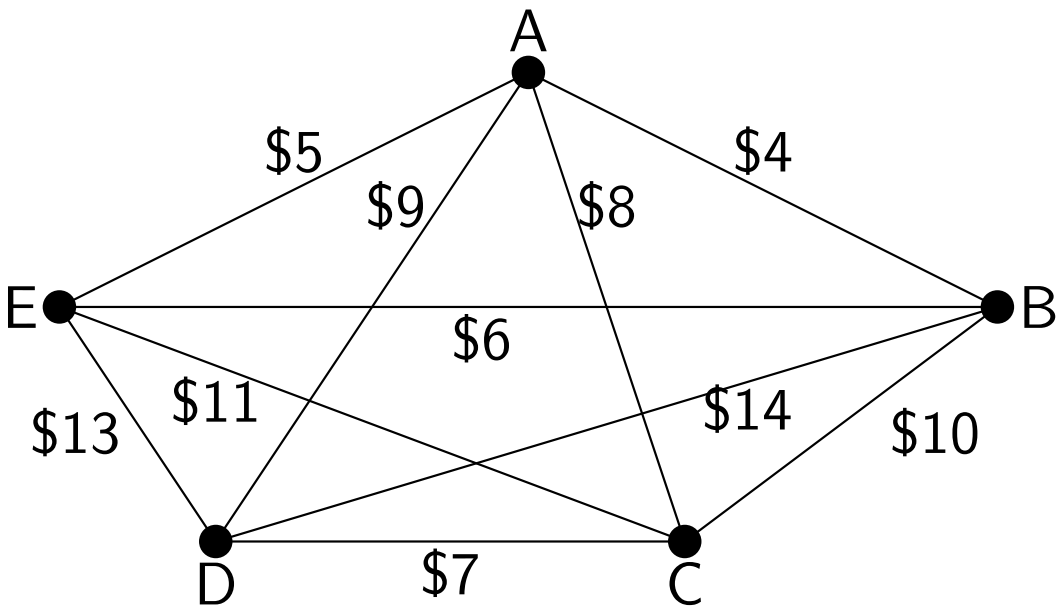
Definition (Minimum Cost Spanning Tree (MCST))

The minimum cost spanning tree is the spanning tree with the smallest total edge weight.

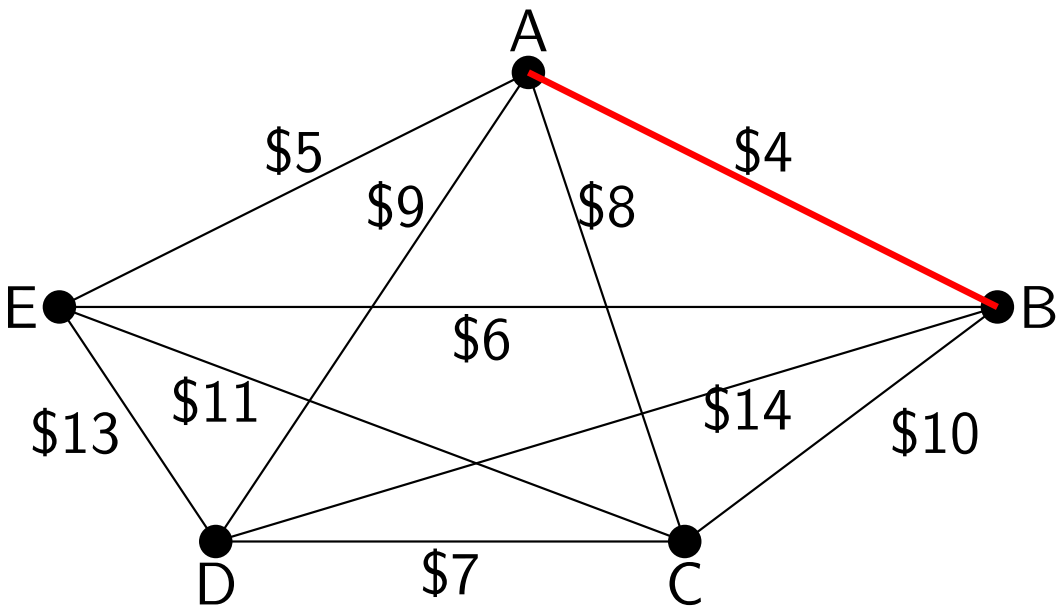
Theorem (Kruskal's Algorithm)

1. *Select the cheapest unused edge in the graph.*
2. *Repeat step 1, adding the cheapest unused edge, unless:*
    1. *adding the edge would create a circuit.*
3. *Repeat until a spanning tree is formed.*

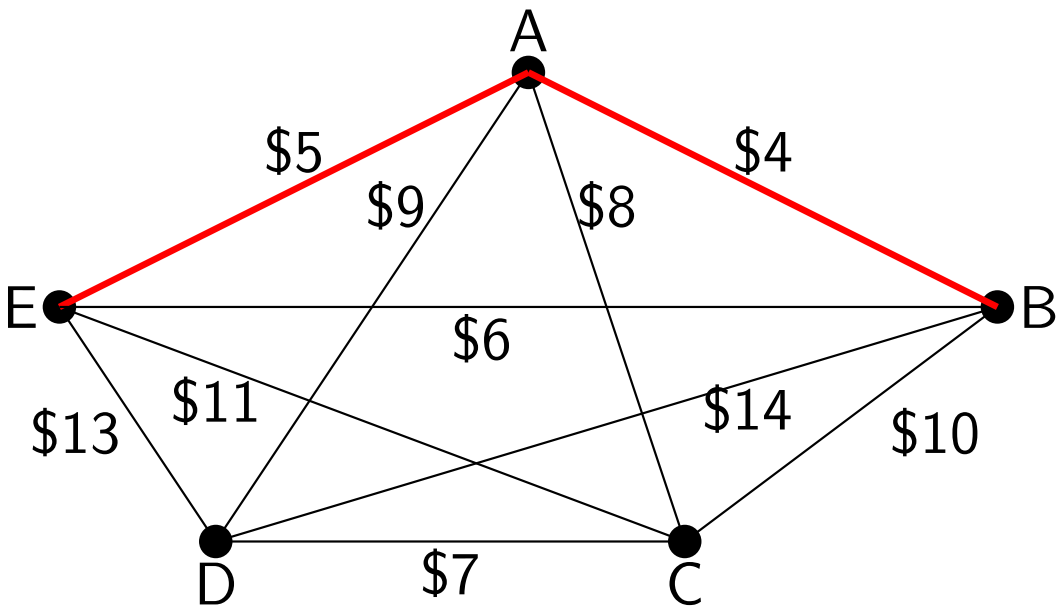
Example



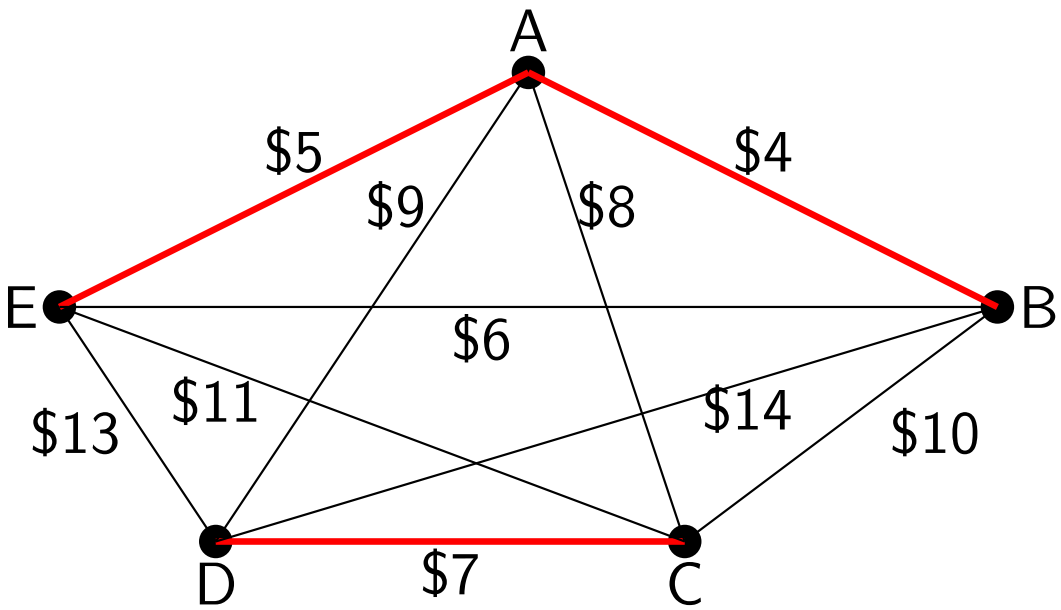
Example



Example



Example



Example