Algebra and Discrete Mathematics
ADM

Bc. Xiaolu Hou, PhD.

FIIT, STU
xiaolu.hou @ stuba.sk

Course Outline

- Vectors and matrices
- System of linear equations
- Matrix inverse and determinants
- Vector spaces and matrix transformations
- Fundamental spaces and decompositions
- Eulerian tours
- Hamiltonian cycles
- Midterm
- Paths and spanning trees
- Trees and networks
- Matching

Recommended reading

- Saoub, K. R. (2017). A tour through graph theory. Chapman and Hall/CRC.
  - Sections 1.1 – 1.5
  - Accessible online (free copy)
  - Alternative download link

Lecture outline

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Eulerian tours

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Graph

Definition

A *graph* consists of two sets: $V(G)$, called the *vertex set*, and $E(G)$, called the edge set. An *edge*, denoted xy, is an unordered pair of vertices $x, y \in V(G)$.

- We will often use G or $G = (V, E)$ as short-hand.
- xy and yx are treated equally, though it is customary to write them in alphabetical order
- Later in the course: direct graphs - the order in which an edge is written provide additional meaning

Graph – example

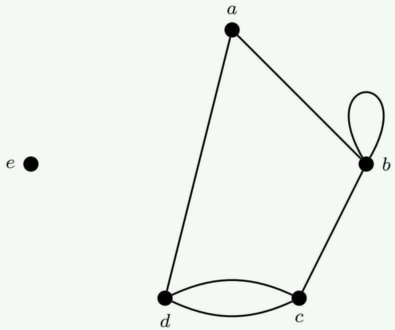
Example

- Let G be a graph where

$$V(G) = \{a, b, c, d, e\},$$

$$E(G) = \{ab, cd, cd, bb, ad, bc\}$$

- Visualization of G: a dot represents a vertex and an edge is a line connecting the two dots (vertices)
- Note: two edges between vertices c and d; a loop at b; no edges at e



Terminologies

Definition

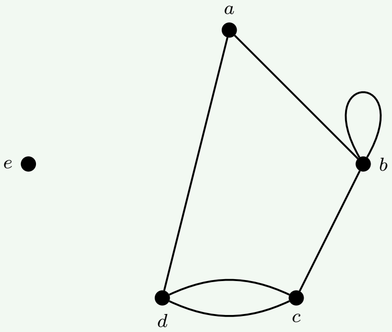
Let $G = (V, E)$ be a graph

- $xy \in E$, x and y are the *endpoints* for edge xy. x (also y) is *incident to* edge xy
- *Adjacent* edges: share an endpoint
- *Adjacent* vertices (*neighbors*): incident to the same edge
- $N(v)$: the set of all neighbors of a vertex v
- *Isolated vertex*: not incident to any edge
- *Loop*: both endpoints of an edge are the same vertex
- *Multi-edges*: more than one edge with the same endpoints
- *Degree* of a vertex: number of edges incident to the vertex, with a loop adding two to the degree. Denoted $\deg(v)$. If the degree is even (resp. odd), the vertex is called *even* (resp. *odd*)

Graph – example

Example

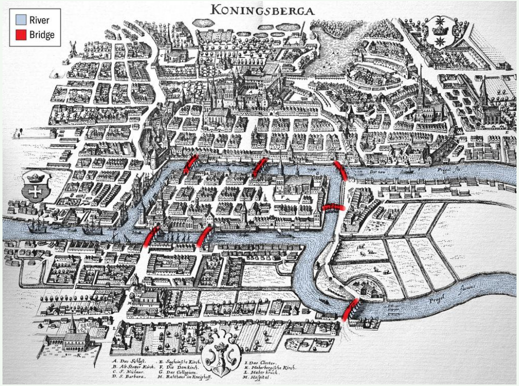
- Edges ab and ad are adjacent
- a and b are adjacent vertices
- $N(d) = \{a, c\}$, $N(b) = \{a, b, c\}$
- e is an isolated vertex
- bb is a loop
- cd is a multi-edge
- $\deg(a) = 2$, $\deg(b) = 4$, $\det(c) = 3$, $\deg(d) = 3$, $\deg(e) = 0$



Modeling with a graph – Königsberg bridge problem

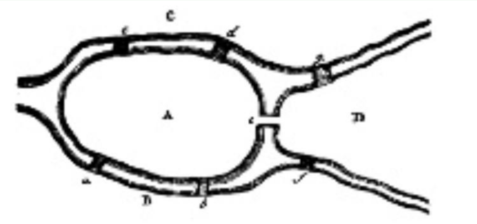
Example

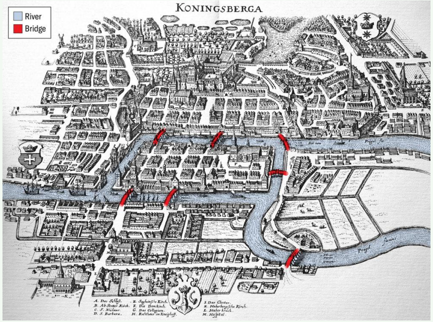
- 1763, Leonhard Euler, one of the greatest mathematicians of all time, published a short paper on the bridges of Königsber, a city in Russia
- Can you leave your home, travel across each of the bridges in the city exactly once, and then return home?
- The puzzle is described as the birth of graph theory



Modeling with a graph – Königsberg bridge problem

Example

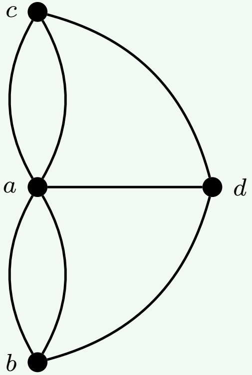


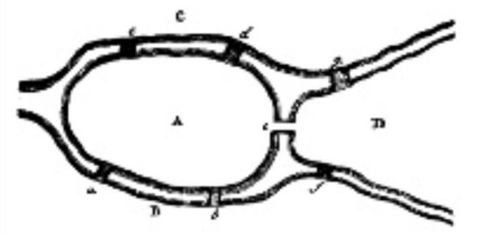


Euler reduced the map of Königsberg to a simpler version where only the relationships between landmasses were of importance

Modeling with a graph – Königsberg bridge problem

Example





We can use a simpler way to model the city with a graph – the vertices represent an island, a south bank, a north bank, and a peninsula $(a, b, c, d$ respectively$)$ – can you leave your home (vertex), travel across each of the bridges (edges) in the city (graph) exactly once and then return home?

Order and size

Definition

- A graph $G = (V(G), E(G))$ is called *finite* if both $V(G)$ and $E(G)$ are finite.
- A graph that is not finite is called an *infinite* graph
- *Order* of G is the number of vertices of G, i.e. $|V|$
- *Size* of G, is the number of edges of G, i.e. $|E|$

Eulerian tours

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Simple graph

Definition

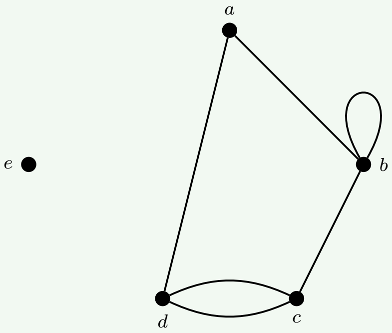
- *Pseudograph*: can contain multi-edges and loops
- *Multipgraph*: can contain multi-edges, but no loops
- *Simple graph*: no multi-edges or loops
- *Trivial graph*: one vertex, no edges

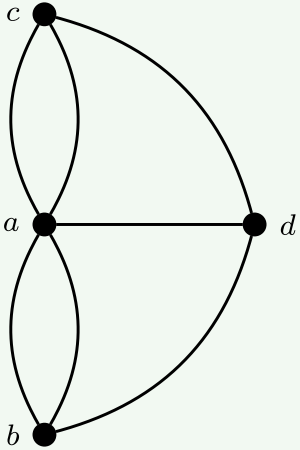
Graphs – example

Example

Multigraph

Pseudograph





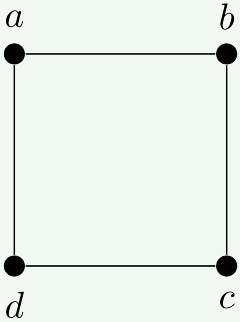
Order: 5, size: 6

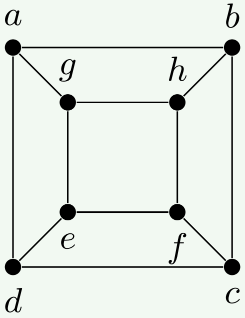
Regular graph

Definition

- *Regular graph*: all of its vertices have the same degree
- $k-$*regular*: all vertices have degree k
- A $3-$regular graph is also called a *cubic* graph

Example





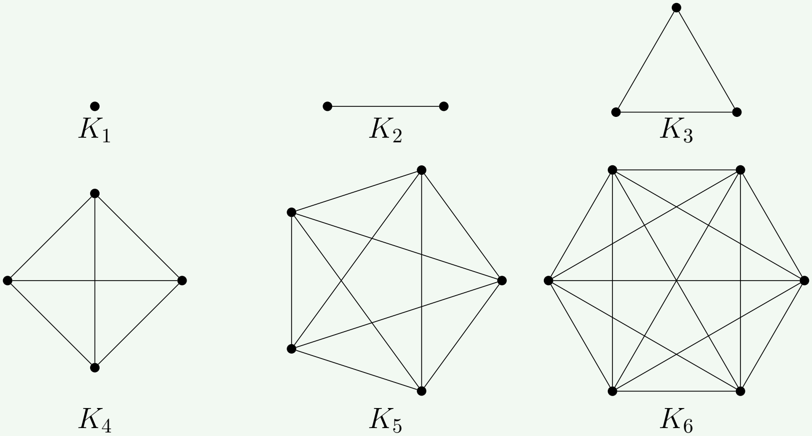
Complete graph

Definition

A simple graph G is *complete* if every pair of distinct vertices is adjacent. The complete graph on n vertices is denoted K_n.

The first six complete graphs

Example



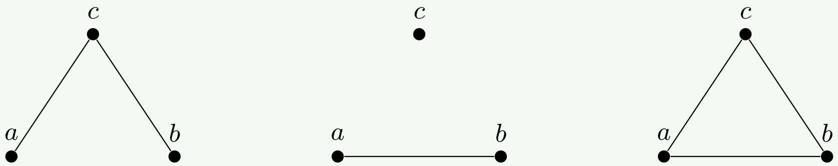
Complement of a graph

Definition

The complement $\overline{G} = (V, \overline{E})$ of a graph $G(V, E)$ is the graph having the same vertex set as G, and its edge set $\overline{E}$ is the complement of E in the set of all unordered pairs of vertices. In other words, for any $u, v \in V$, $uv \in \overline{E}$ iff $uv \notin E$.

- G and $\overline{G}$ together forms a complete graph

Example



Bipartite graphs

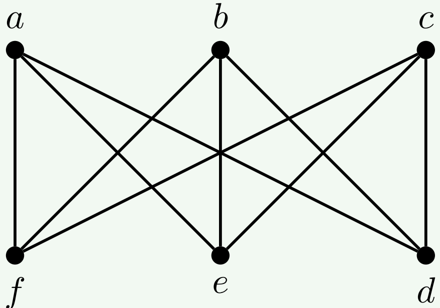
Definition

A graph $G = (V, E)$ is *bipartite* if the vertices can be partitioned into two sets, V_1 and V_2, so that

$$V_1 \cap V_2 = \emptyset, \quad V = V_1 \cup V_2,$$

and every edge has exactly one endpoint in V_1 and the other endpoint in V_2.

Example



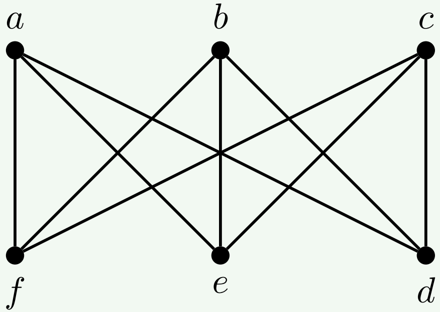
Complete bipartite graph

Definition

A simple bipartite graph $G = (V_1 \cup V_2, E)$ is a *complete bipartite graph* if every vertex in V_1 is adjacent to every vertex in V_2. If $|V_1| = m$, $|V_2| = n$, we write $K_{m,n}$

Example

$K_{3,3}$



Eulerian tours

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Definitions

Definition

Let G be a graph

- *Walk*: a sequence of vertices so that there is an edge between consecutive vertices. A walk can repeat vertices and edges.
- *Trail*: a walk with no repeated edges
- *Path*: a trial with no repeated vertices. A path on n vertices is denoted P_n
- *Closed walk*: a walk that starts and ends at the same vertex
- *Circuit*: a closed trial
- *Cycle*: a closed path. a cycle on n vertices is denoted C_n

Length of any of these tours is the number of edges.

- Path $\subseteq$ Trail $\subseteq$ Walk
- Cycle $\subseteq$ Circuit $\subseteq$ Closed walk

Length of a tour – example

Example

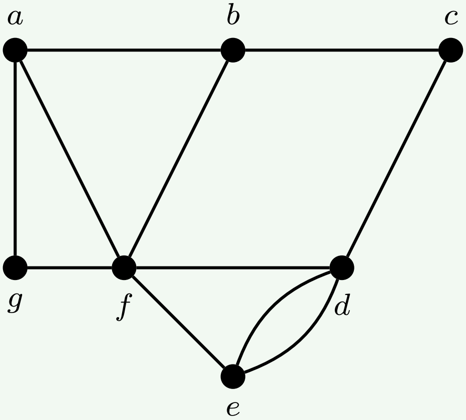
- P_n has length $n - 1$
- C_n has length n

Touring a graph – example

Example

Given the graph, find

- a trial (that is not a path) from a to c
- a path from a to c
- a circuit (that is not a cycle) starting at b
- a cycle starting at b



Touring a graph – example

Example

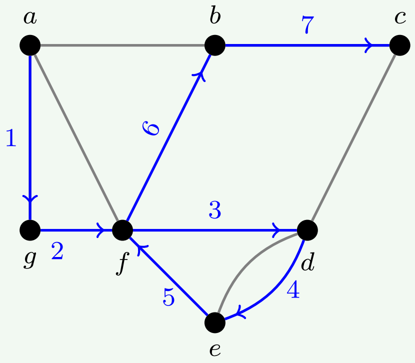


Figure: A trial from a to c

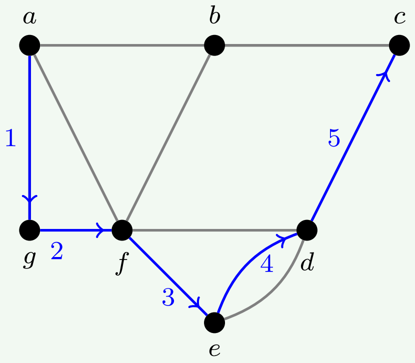


Figure: A path from a to c

Touring a graph – example

Example

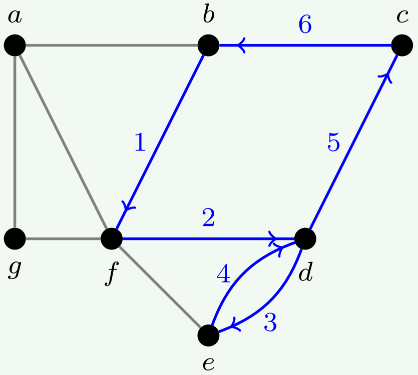


Figure: A circuit starting at b

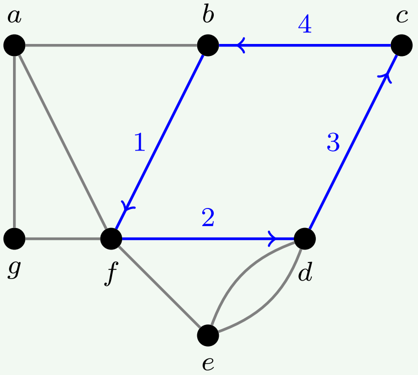


Figure: A circle starting at b

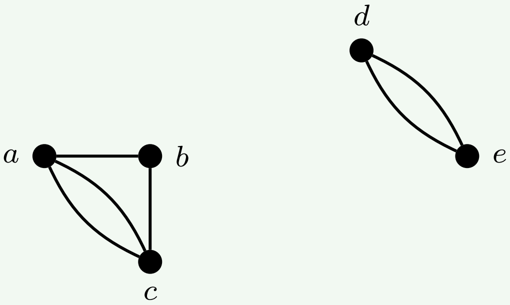
Connected vertices

Definition

Let G be a graph. Two vertices x and y are *connected* if there exists a path from x to y in G. The graph G is *connected* if every pair of distinct vertices is connected.

Example

Not connected



Eulerian circuit

Definition

Let G be a graph. An *Eulerian circuit* (or *trail*) is a circuit (or trail) that contains every edge and every vertex of G. If G contains an Euclidean circuit it is called *Eulerian* and if G contains an Eulerian trail but not an Eulerian circuit it is called *semi-Eulerian*.

Eulerian graph

Theorem

A graph G is Eulerian if and only if

- *G is connected and*
- *every vertex has an even degree*

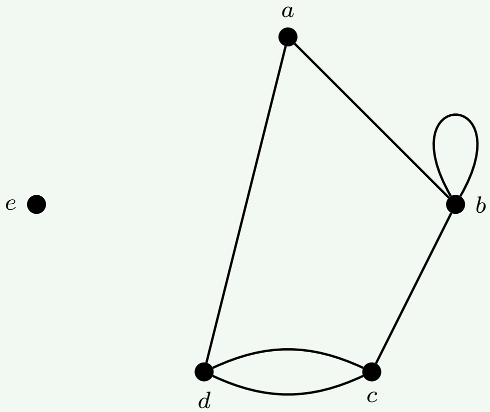
A graph G is semi-Eulerian if and only if

- *G is connected and*
- *exactly two vertices have odd degree*

- When traveling through a graph, we need to pair each entry edge with an exit edge.
- If a vertex is odd, then there is no pairing available and we would eventually get stuck at that vertex
- If the starting and ending locations are different, then exactly two vertices must be odd since the first edge out of the starting vertex does not need to be paired with a return edge and the last edge to the ending vertex does not need to be paired with an exit edge

Eulerian and semi-Eulerian graphs – example

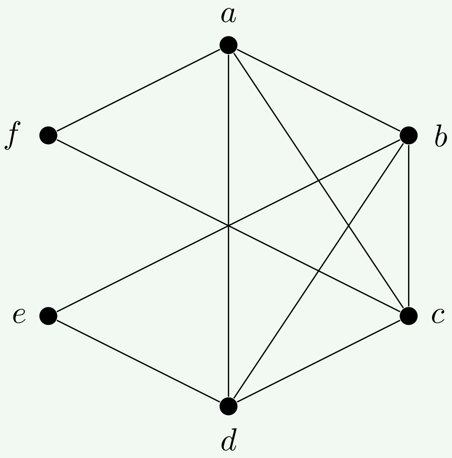
Example



Not connected – not Eulerian, not semi-Eulerian. The graph has two *components*.

Eulerian and semi-Eulerian graphs – example

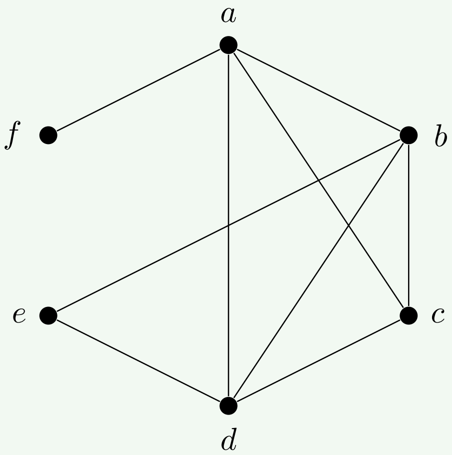
Example



Eulerian – all vertices have degree 2 or 4

Eulerian and semi-Eulerian graphs – example

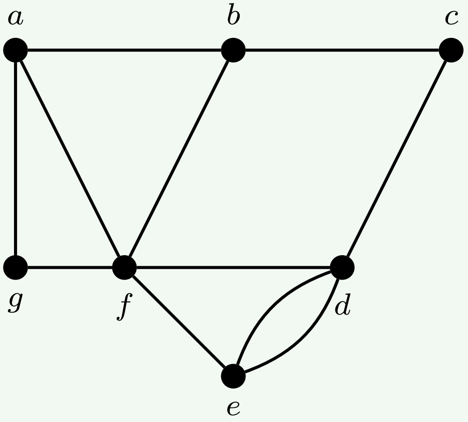
Example



semi-Eulerian – exactly two vertices are odd

Eulerian and semi-Eulerian graphs – example

Example



Connected, but not Eulerian or semi-Eulerian - more than two odd vertices (a, b, e, f)

Handshaking Lemma

Theorem (Handshaking Lemma)

Let $G = (V, E)$ *be a graph. Suppose* $V = \{v_1, v_2, \ldots, v_n\}$*, then*

$$\deg(v_1) + \deg(v_2) + \cdots + \deg(v_n) = 2|E|$$

Proof.

Since each edge contributes two to the sum of degrees, one at each of the two endpoints, the sum of degrees is even and equal to twice the number of edges. $\square$

Corollary

Corollary

In any graph, there is an even number of odd vertices.

Proof.

$$\sum_{v_i \text{ has an odd degree}} \det(v_i) + \sum_{v_j \text{ has an even degree}} \deg(v_j) = 2|E|$$

The second term on the left is even, the sum is even. $\square$

Eulerian tours

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Fleury's Algorithm

- Find an Eulerian circuit or an Eulerian trail
- By the previous theorem, the input graph must be connected and has zero or two odd vertices

Fleury's Algorithm

1. Choose a starting vertex:
   - If the graph G has no vertices of odd degree, start from any vertex
   - If G has exactly two vertices of odd degree, start from one of them
2. Choose an edge to traverse: Select an edge incident to the current vertex v, ensuring the graph containing remaining edges is connected after removing it
3. Traverse and remove the edge: Move to the other endpoint of the selected edge, then delete the edge from the graph. The new vertex becomes the current vertex.
4. Repeat steps 2 and 3 until there are no more edges left in the graph

Output

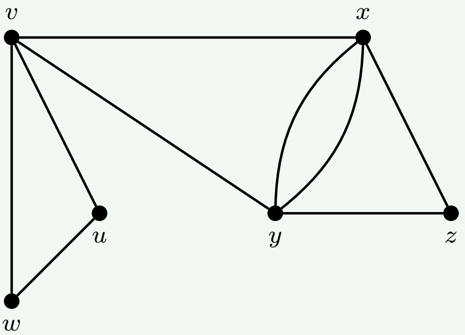
- If G originally had no odd-degree vertices, Eulerian circuit
- If G had exactly two odd-degree vertices, Eulerian trail

Practical Implementation: Maintain two copies of the graph:

- One for tracking the Eulerian path or cycle.
- Another for dynamically removing edges

Fleury's Algorithm – example

Example



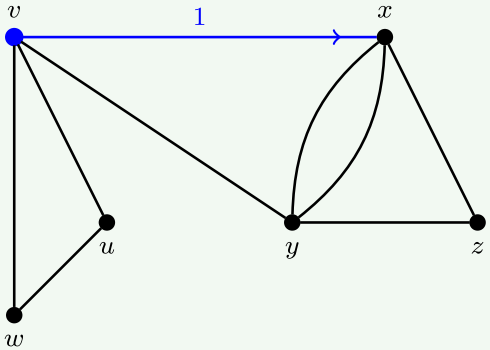
The graph is Eulerian:

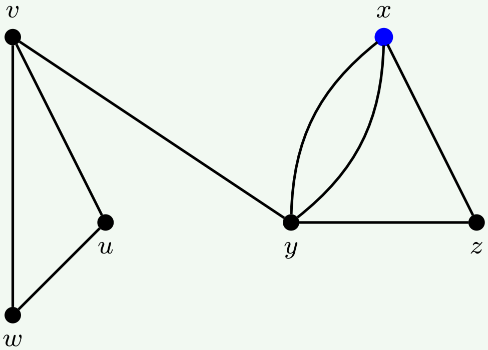
$\deg(u) = 2, \quad \deg(v) = 4, \quad \deg(w) = 2, \quad \deg(x) = 4, \quad \deg(y) = 4, \quad \deg(z) = 2$

- Step 1. start vertex v
- Step 2. choose edge vx

Fleury's Algorithm – example

Example

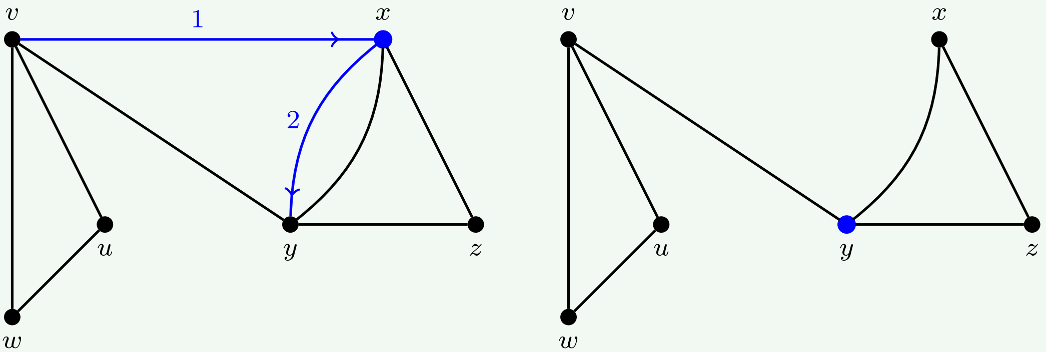




- Step 3. travel to vertex x and delete edge vx. Current vertex: x.

Fleury's Algorithm – example

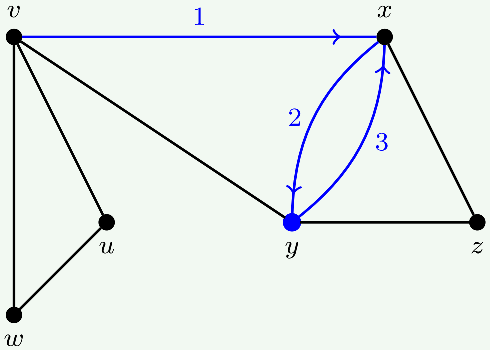
Example

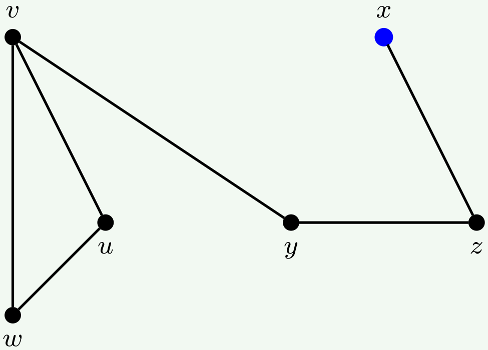


- Step 2. choose edge xy
- Step 3. travel to vertex y and delete edge xy. Current vertex: y.

Fleury's Algorithm – example

Example

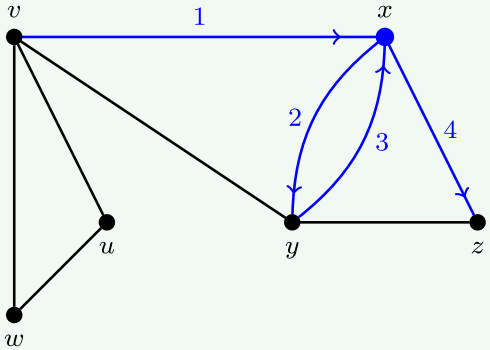


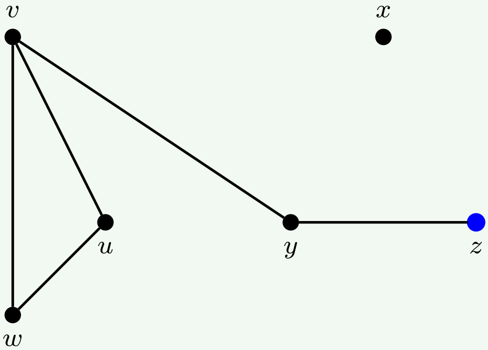


- Step 2. choose edge yx, note: cannot choose edge yv, its removal would disconnect the remaining graph
- Step 3. travel to vertex x and delete edge yx. Current vertex: x.

Fleury's Algorithm – example

Example

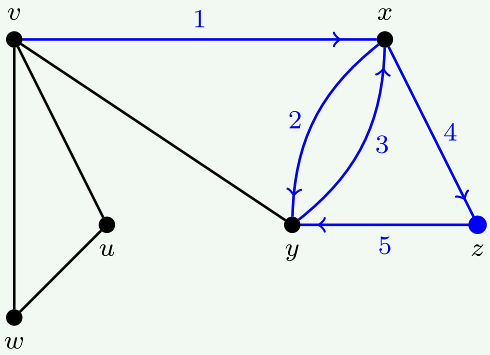


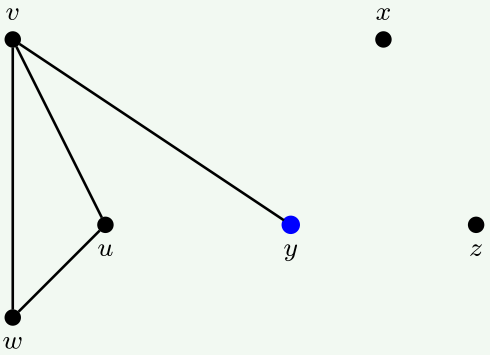


- Step 2. only one available edge xz
- Step 3. travel to vertex z and delete edge xz. Current vertex: z.

Fleury's Algorithm – example

Example

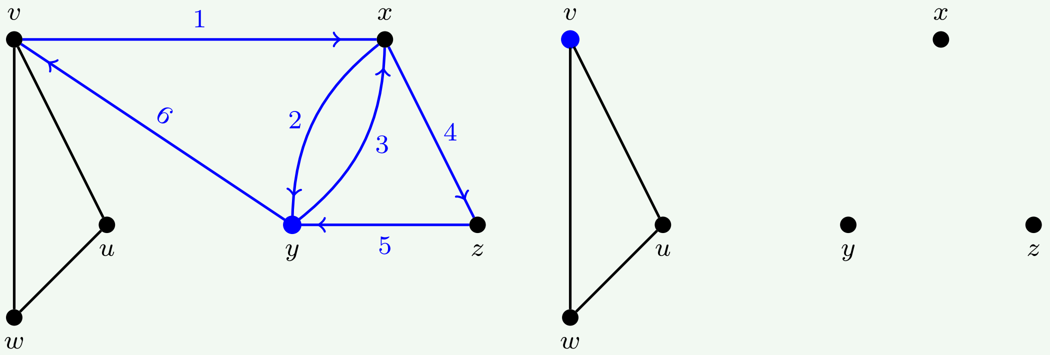




- Step 2. only one available edge zy
- Step 3. travel to vertex y and delete edge zy. Current vertex: y.

Fleury's Algorithm – example

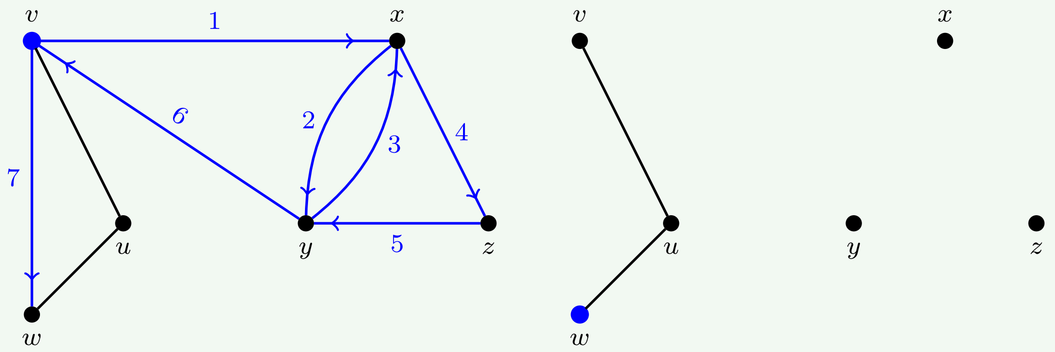
Example



- Step 2. only one available edge yv
- Step 3. travel to vertex v and delete edge yv. Current vertex: v.

Fleury's Algorithm – example

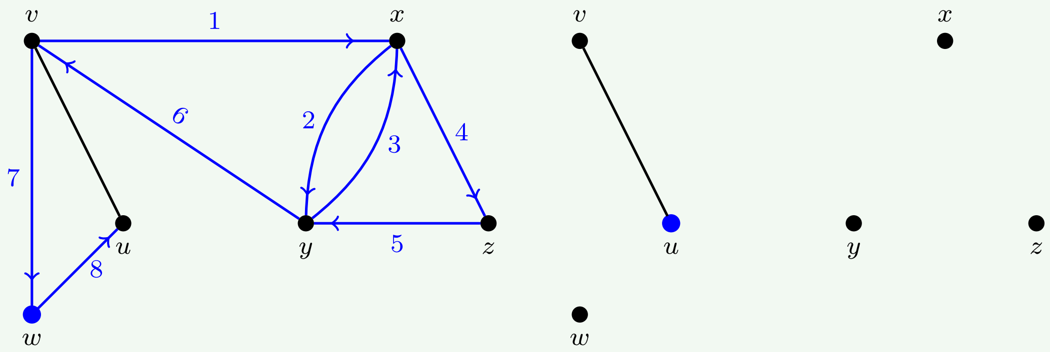
Example



- Step 2. both vu and vw can be chosen. Here we choose vw.
- Step 3. travel to vertex w and delete edge vw. Current vertex: w.

Fleury's Algorithm – example

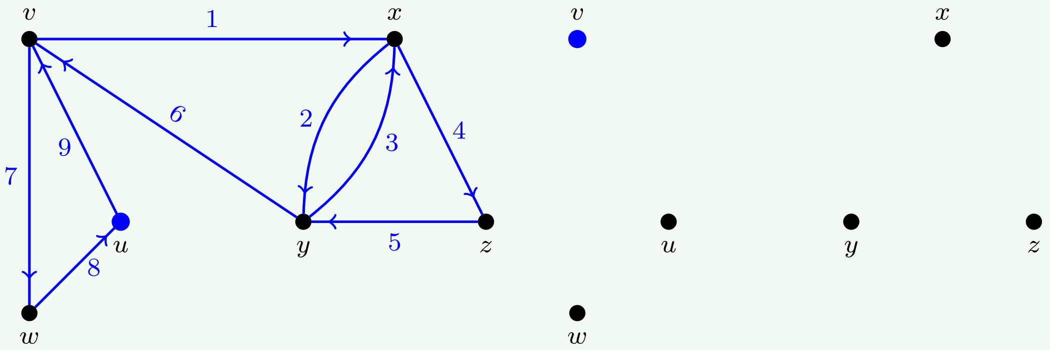
Example



- Step 2. only one available edge wu
- Step 3. travel to vertex u and delete edge wu. Current vertex: u.

Fleury's Algorithm – example

Example



- Step 2. only one available edge uv
- Step 3. travel to vertex v and delete edge uv
- We have obtained an Eulerian circuit starting and ending at vertex v

Hierholzer's Algorithm

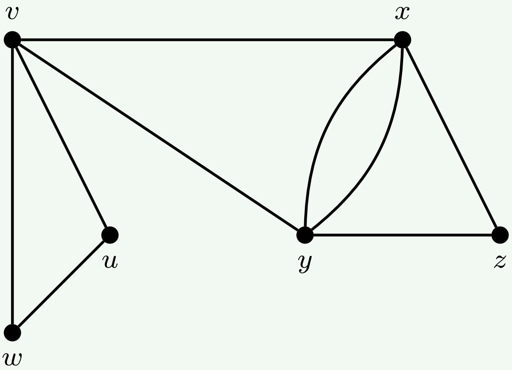
- Find an Eulerian circuit
- Input: an Eulerian graph G – connected, all vertices are even

Hierholzer's Algorithm

1 Choose a starting vertex, say v, and find an initial circuit C starting at v.
2 If there exists a vertex x in the current circuit C that has unused edges, form a new circuit C' starting at x that uses two of those unused edges
3 Merge the newly found circuit C' into the existing circuit C, updating C accordingly.
4 Repeat steps 2 and 3 until all edges of G have been traversed.

Hierholzer's Algorithm – example

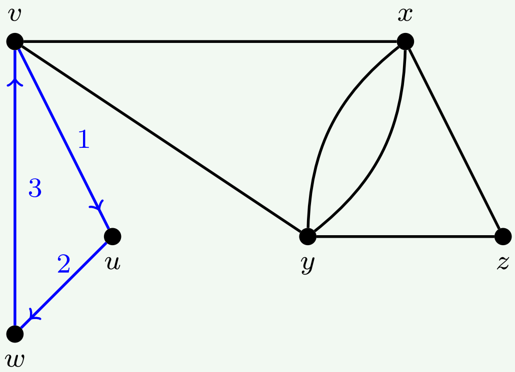
Example



Same graph, we have discussed that the graph is Eulerian.

Hierholzer's Algorithm – example

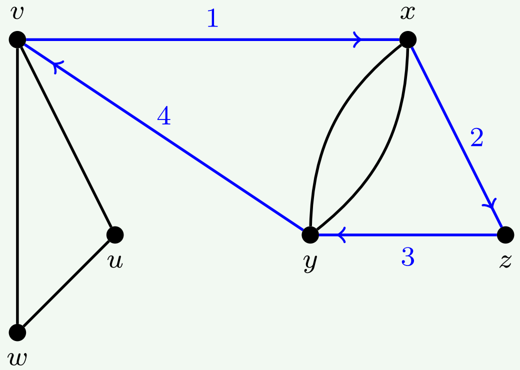
Example



- Step 1: choose vertex v and find a circuit starting at v

Hierholzer's Algorithm – example

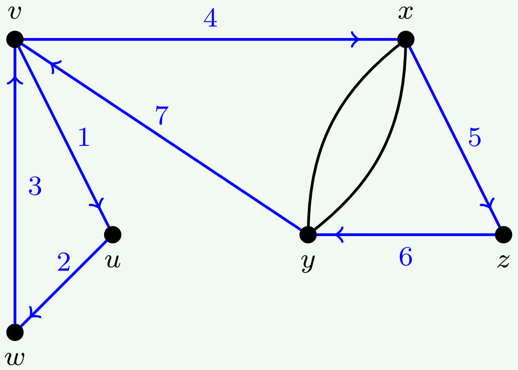
Example



- Step 2: since $\det(v) = 4$, two edges remain from v. We can find a second circuit starting at v.

Hierholzer's Algorithm – example

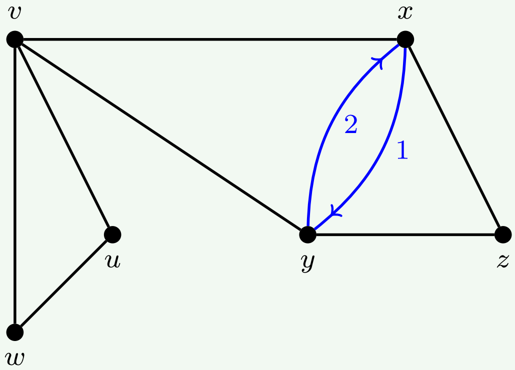
Example



- Step 3: combine the two circuits from Step 1 and Step 2. There are multiple ways to combine two circuits.

Hierholzer's Algorithm – example

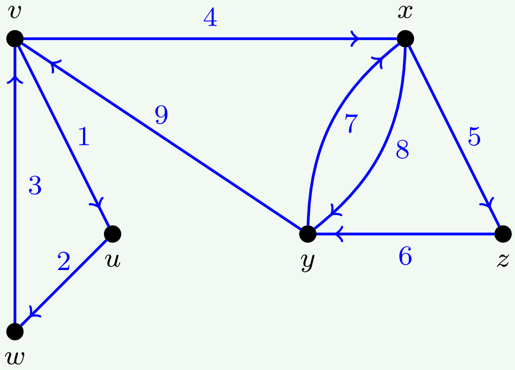
Example



- Step 2: $\deg(x) = 4$ and two edges remain for x, we can find a circuit starting from x

Hierholzer's Algorithm – example

Example



- Step 3: combine the new circuit with the existing one
- We have obtained an Eulerian circuit

Eulerian tours

- Definitions and terminologies
- Different types of graphs
- Touring a graph
- Eulerian circuit algorithms
- Eulerization

Eulerization – definition

Definition

Given a connected graph $G = (V, E)$, an *Eulerization* of G is the graph $G' = (V, E')$ so that

- G' is obtained by duplicating edges of G, and
- every vertex of G' is even

A *semi-Eulerization* of G results in a graph G' so that

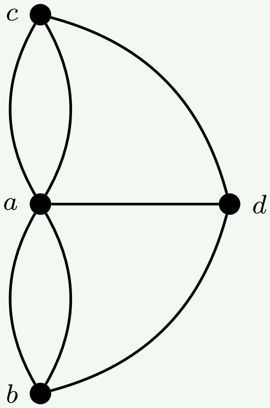
- G' is obtained by duplicating edges of G, and
- exactly two vertices of G' are odd

Optimal exhaustive tour

- In the context of the Königsberg Bridge Problem, duplicating an edge would be equivalent to walking the same bridge twice
- Although this solution would be outside the original parameters (walking each bridge exactly once), it allows for an approximate solution using the fewest number of duplications.
- This is referred to as finding an *optimal exhaustive tour* of a graph

The Königsberg Bridge Problem

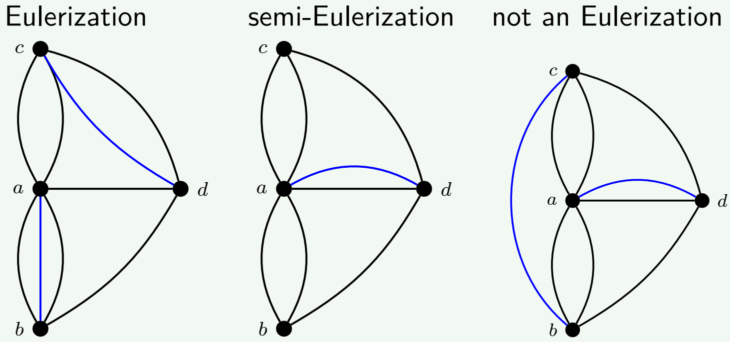
Example



We have discussed that the graph modeling Königsberg city is not Eulerian or semi-Eulerian

The Königsberg Bridge problem

Example



- Note that when creating the new graph, edges must be duplicated, not added
- Of course edges can be added in a city, but unrealistic from the standpoint of a person touring the bridges of a city during a specific moment in time. Hence, we only allow duplications of edges

The optimization question

- We are interested in the question: how can we Eulerize (or semi-Eulerize) a graph using the fewest number of edge duplications?
- We are attempting to find an optimal tour of the graph, that is minimize the total length of the circuit (or trail).

Eulerization method

1. Identify the odd vertices of the graph
2. Pair up the odd vertices, trying to pair as many adjacent vertices as possible while also avoiding paring vertices far away from each other[1]
3. Duplicate the edges along an optimal path from one vertex to its pair

Note:

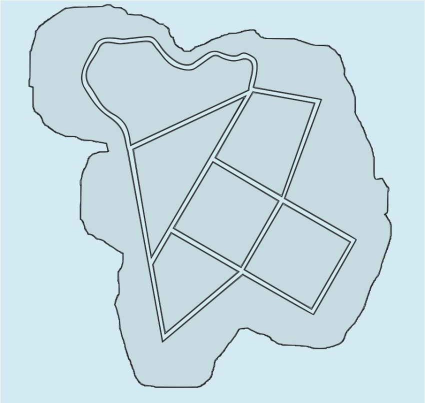
- In the process of determining which edges to duplicate along optimal paths, never repeat an edge more than once
- If an edge is crossed three times, removing two of the duplications will not change the parity of the endpoint of the edge
  - that is, a vertex will remain odd or remain even when subtracting two from the degree

[1] Distance: the length of the shortest path connecting those two vertices.

Eulerization – example

Example

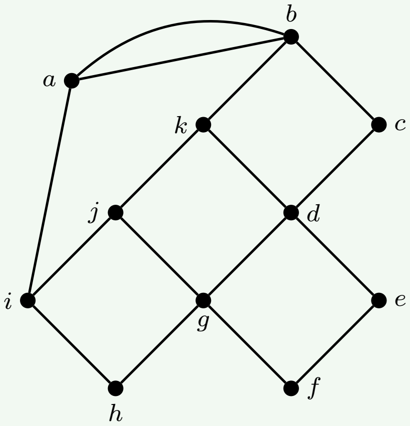
- The citizens of the small island town of Sunset Island want to hire a night patrol during the busy summer tourist season
- Model the town as a graph and find an optimal Eulerization of the graph

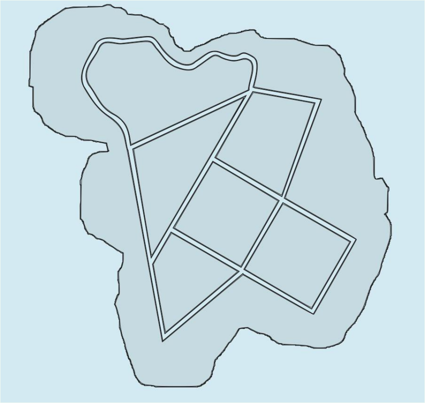


Eulerization – example

Example

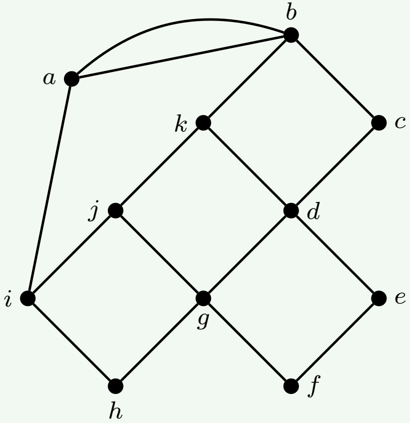
The graph modeling - vertices represent intersections and edges represent street blocks

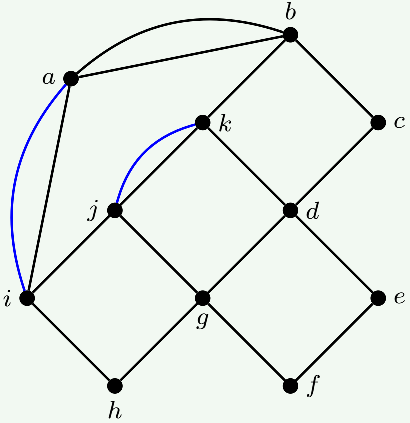




Example

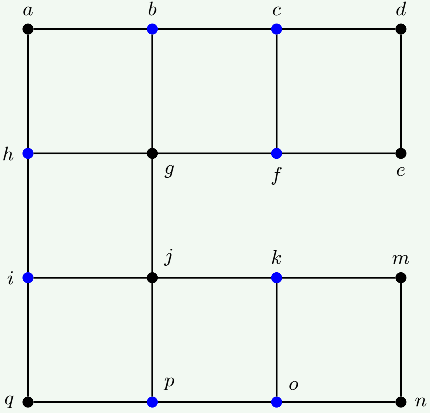
- Four odd vertices: a, i, j, k
- They can be split into two pairs of adjacent vertices: a, i and j, k

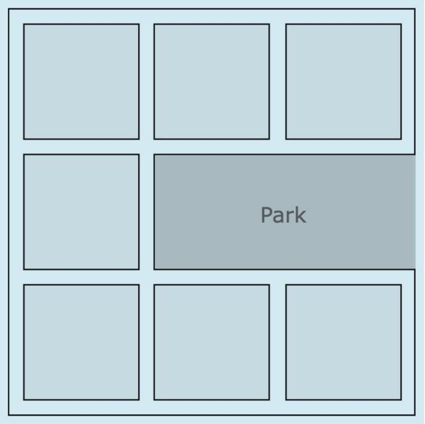




Example

- Task: put up fliers along each block of a small portion of a town
- Requirement: travel along each street once but cannot put up fliers in the park



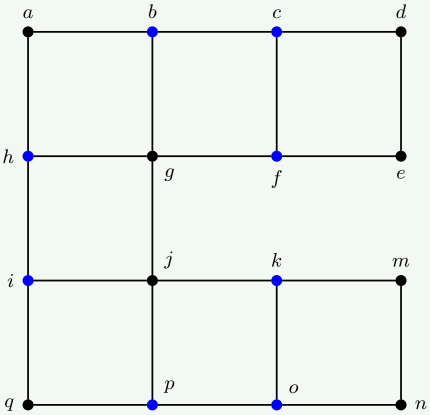


We model the graph and identify the odd vertices (in blue)

Eulerization – example

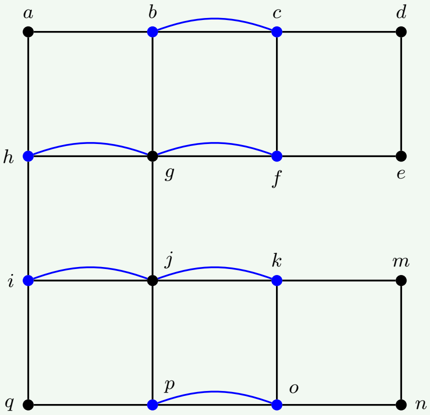
Example

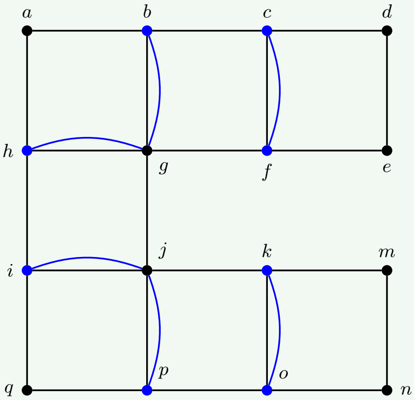
- Unlike the previous example, we cannot split all the 8 odd vertices into adjacent pairs, which corresponds to only 4 duplicated edges
- 5 duplicated edges is also not possible
  - if we use exactly 2 adjacent pairs, we are left with 2 pairs each of which is of distance 2 apart and thus requiring another 4 edges duplications
  - if we use 3 adjacent pairs, we are left with 1 pair of distance 3 apart, which requires 3 more edges
- At least 6 edge duplications are needed



Eulerization – example

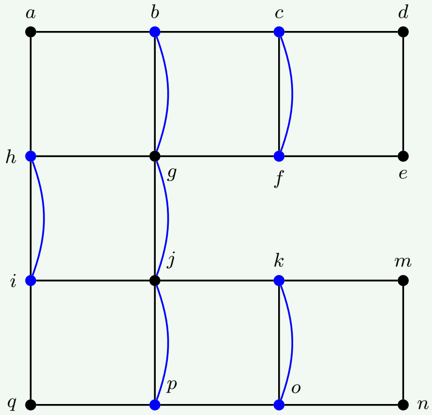
Example

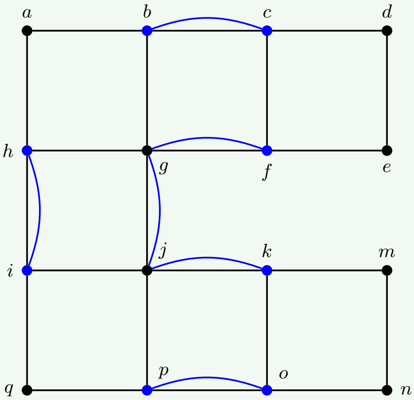




Eulerization – example

Example





Weighted graph

- In the previous examples, the streets follow a highly structured grid layout, where traveling along one block is comparable to traveling along another.
- What happens when different streets require varying levels of effort to traverse?

Definition

A *weighted graph* $G = (V, E, \omega)$ is a graph where each of the edges has a real number associated with it. This number is referred to as the *weight* and denoted $\omega(xy)$ for an edge xy

Remark

- Wight of an edge can represent: length, time, cost, etc.
- A weighted graph can also refer to a graph in which each of the vertices is assigned a weight

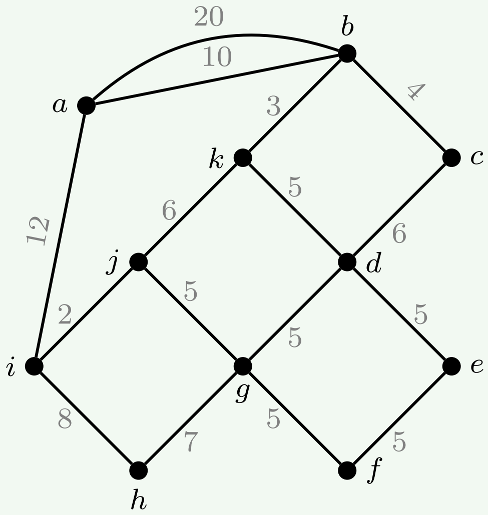
Chinese Postman Problem

- The weighted version of an Eulerization problem is called the *Chinese Postman Problem*
- The name originates not from anything particular about postmen in China, but rather from the mathematician who first proposed the problem —the Chinese mathematician Mei-Ku Kwan (管梅谷) in 1962

Chinese Postman Problem – example

Example

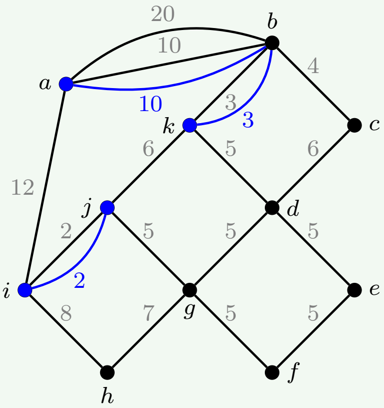
- Suppose the travel time through the streets of Sunset Island varies depending on the street.
- Find an optimal Eulerization taking into account the weights

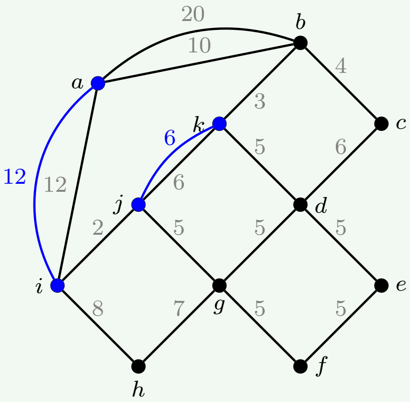


Chinese Postman Problem – example

Example

The previous Eulerization we have obtained (on the right) is not optimal (two duplicated edges total 18). A better Eulerization duplicates three edges for a total of 15





Remarks

- Be careful when duplicating edges in a graph with multi-edges, indicating which edge has been duplicated – including the edge weight clarifies which option was used.
- In general, solving the Chinese Postman Problem can be quite challenging
- Most small examples can be solved by inspection
- The choice of which edges to duplicate when working with a weighted graph relies in part on shortest paths between two vertices.
- The difficulty is in choosing which vertices to pair.
- This will be discussed later in the course

About tutorials

- Enrichment Questions
  - These questions are intended to deepen your understanding.
  - Will *NOT* appear in the exams
  - Participation is optional and for your interest.
  - If you'd like to discuss them, feel free to email me or your tutor (cvičiaci) to arrange a meeting.