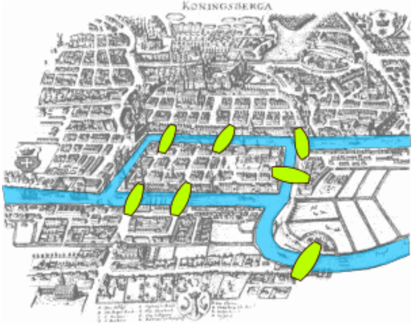
Eulerian Cycles

Seven Bridges of Königsberg

Königsberg (now Kaliningrad, Russia) around 1735

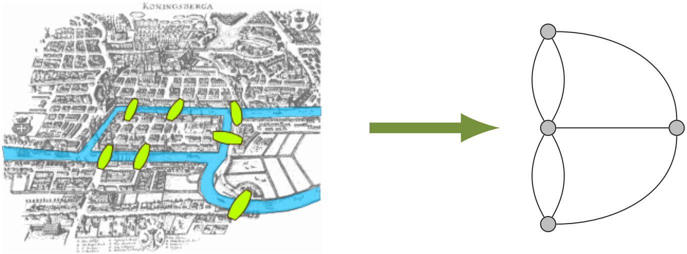


Problem

- Find a walk through the city that would cross each bridge once and only once.

EULER's Solution

Represent problem as graph



Original Problem

- Find a walk through the city that would cross each bridge once and only once.

Gaph Problem

- Does G contain a path using each edge exactly once?

EULER's Solution

Given a graph $G = (V, E)$ with $|E| = m$ and index arithmetic modulo m.

Eulerian Cycle

An *Eulerian cycle* in a graph G is a sequence of vertices and edges $\langle v_1, e_1, \ldots, v_m, e_m, v_1 \rangle$ such that $e_i = v_i v_{i+1}$ and $e_i \neq e_j \Leftrightarrow i \neq j$.

Eulerian Path

An *Eulerian path* in a graph G is a sequence of vertices and edges $\langle v_1, e_1, \ldots, v_m, e_m, v_{m+1} \rangle$ such that $e_i = v_i v_{i+1}$ and $e_i \neq e_j \Leftrightarrow i \neq j$.

In an Eulerian cycle the start and end vertex hast to be equal. In an Eulerian path they can be different.

EULER's Solution

Theorem

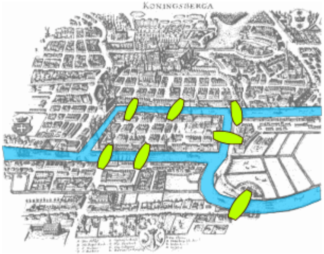
A connected graph has an Eulerian cycle if and only if the degree of all vertices is even, i. e., $\forall\, v \in V \;\; \exists\, k \in \mathbb{N}\colon\; \deg(v) = 2k$.

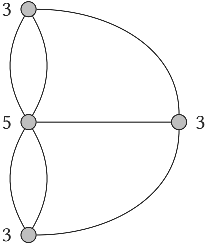
Theorem

A graph has an Eulerian path if and only if it is connected and has at most two vertices with an odd degree.

Seven bridges of Königsberg

Degree of all vertices is odd.



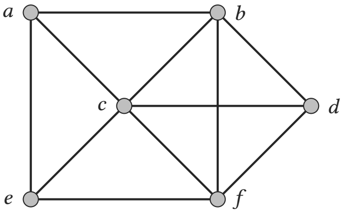


Thus, there is no walk through the city that would cross each bridge once and only once.

Algorithm

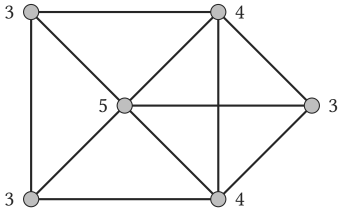
Algorithm

Given a graph G. Does G have an Eulerian path?
Algorithm based on HIERHOLZER's algorithm from 1837.



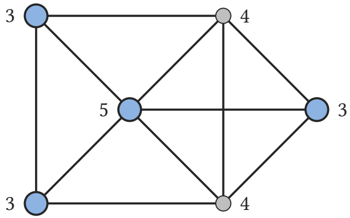
Algorithm

1. Determine the degree of all vertices.



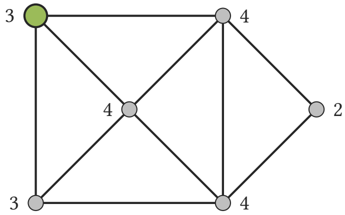
Algorithm

2. **IF** There are more than two vertices with an odd degree **THEN** **STOP**, G has no Eulerian path.

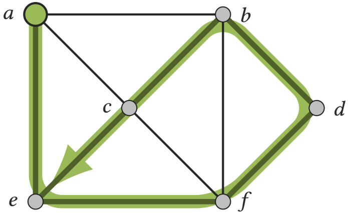


Algorithm

3. Choose any starting vertex v. Select a vertex with odd degree if possible.



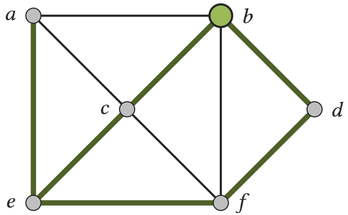
4. Follow a trail T of edges from v until getting stuck.



$T = \langle a, e, f, d, b, c, e \rangle$

Algorithm

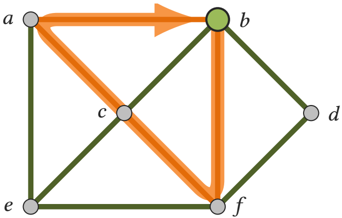
5. **While** There is a vertex v belonging to the current trail T but has adjacent edges not part of T.



$T = \langle a, e, f, d, b, c, e \rangle$

Algorithm

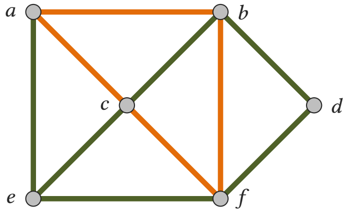
5.1 Start another trail T' from v, following unused edges until returning to v.



$T = \langle a, e, f, d, b, c, e \rangle \quad T' = \langle b, f, c, a, b \rangle$

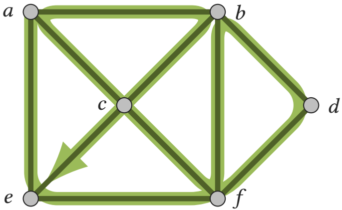
Algorithm

5.1 Start another trail T' from v, following unused edges until returning to v.



$T = \langle a, e, f, d, b, c, e \rangle \quad T' = \langle b, f, c, a, b \rangle$

5.2 Join the tour T' formed in this way to the previous trail T.



$T = \langle a, e, f, d, b, f, c, a, b, c, e \rangle$

Complete Algorithm

If There are more than two vertices with an odd degree **Then**
  Stop. G has no Eulerian path.
Choose a starting vertex v, if possible, with odd degree.
Follow a trail T of edges from v until getting stuck.
While There is a vertex v belonging to the current trail T but has adjacent edges not part of t.
  Start another trail T' from v following unused edges until returning to v.
  Join the tour T' formed in this way to the previous trail T.

This algorithm can be implemented in $O(m)$ time.