GRAPH THEORY and APPLICATIONS

Euler Tours

and

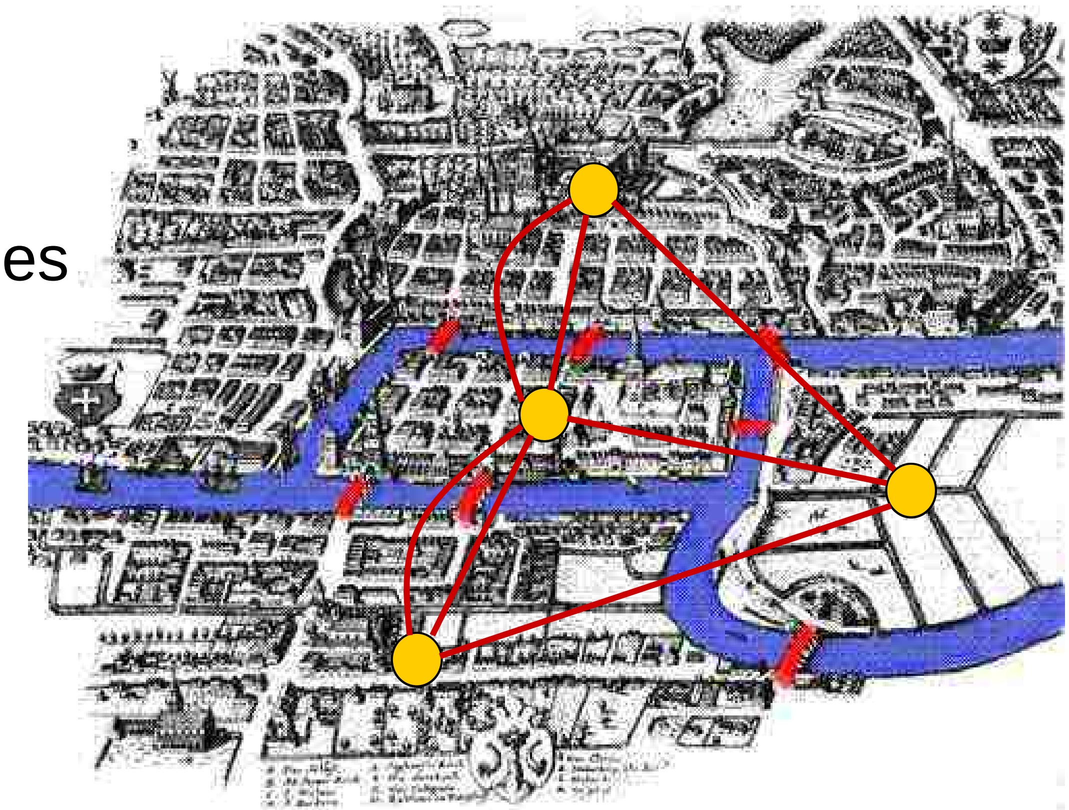
Hamilton Cycles

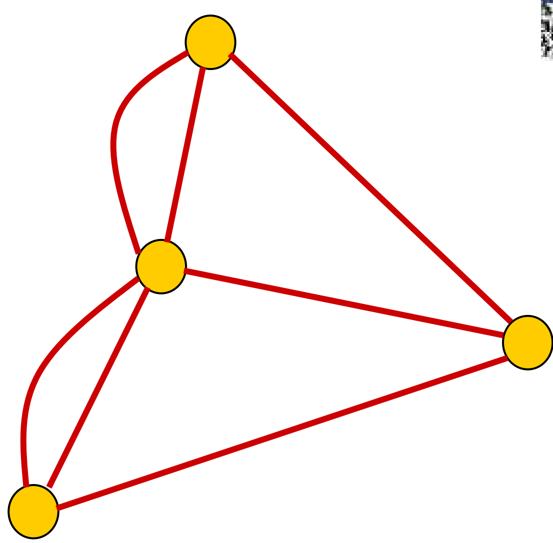
Euler Tour

- Euler trail: A trail that traverses every edge of a graph
- Earliest known paper on graph theory:
  - Euler, L., *Solutio problematis ad geometriam situs pertinentis*. Comment. Academia Sci. I. Petropolitanae, 8, 128-140, 1736.
- Euler showed that it was impossible to cross each of the seven bridges of Koningsberg once and only once during a walk through the town.

Koningsberg at that time

- Father of graph theory, Euler
  - Konigsberg bridges problem (1736)





Euler Tour

- A **tour** of G: A closed walk that traverses each edge of G at least once.
- **Euler tour**: A tour which traverses each edge exactly once.
  ≡ A closed Euler trail.
- A graph is **Eulerian**, if it contains an Euler tour.

An example problem

A postman delivers mail every day in a network of streets.

- To minimize his journey he wishes to know whether it is possible to:
  - traverse this network and return to his depot
  - without walking any street more than once
- Solution to this problem is finding an Eulerian tour of the corresponding graph.

Eulerian graphs

Theorem: An undirected nonempty graph is eulerian *(or has an Euler trail),* iff it is connected and the number of vertices with odd degree is 0 *(or 2).*

The proof of this theorem is useful to understand how to construct Euler trails on any graph.

Proof

The conditions are necessary, because:

- If an Euler trail exists then:
  - G must be connected
  - Only the vertices at the ends of an Euler trail can be of odd degree.

Now, show the conditions are sufficient:

- The theorem is true for $|E| = 2$
- Let G have $|E| > 2$, satisfy the conditions.
- If G contains two vertices of odd degree, denote them by v_1 and v_2.

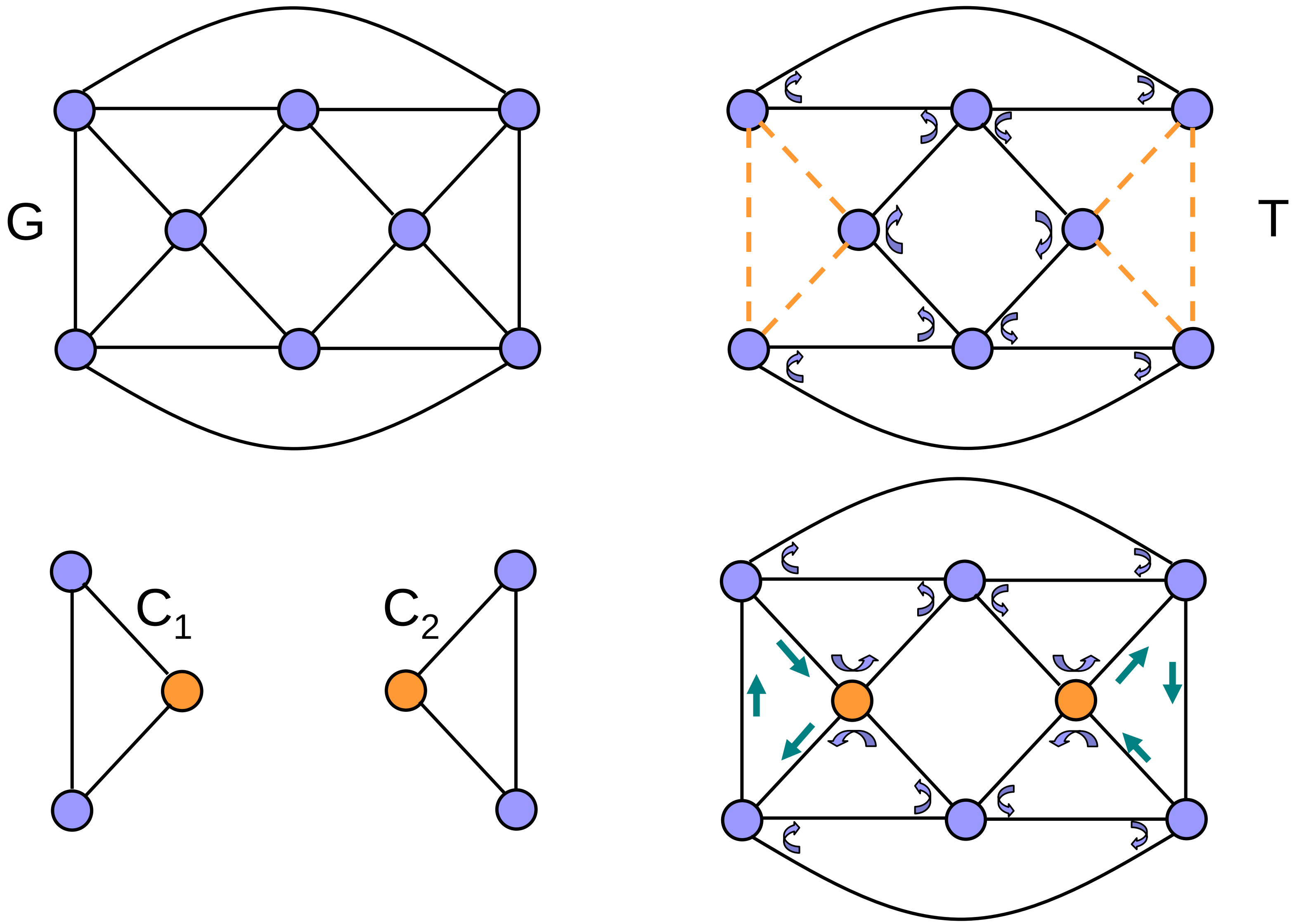
Proof – 2

- Consider tracing a tour T from vertex v_i
  - $v_i = v_1$ if there are vertices of odd degree.
- Trace T leaving each new vertex by an unused edge until a vertex v_j is encountered for which every incident edge has been used.
- If G contains no vertices of odd degree then:
  - $v_j = v_i$
- Otherwise:
  - $v_j = v_2$

Proof – 3

- Suppose T doesn't use every edge of G.
- Remove all used edges from G.
- Then, we are left with a subgraph G'.
- G' :
  - is not necessarily connected.
  - contains vertices of even degree.
- By induction, each component of G' contains an Euler tour.
- G is connected $\Rightarrow$ T must pass through at least one vertex in each component of G'.

Constructing an Euler trail



Euler trail in digraphs

Corollary:

- A directed graph is eulerian iff it is connected, and is balanced.
- A digraph has an euler trail iff it is connected, and the degrees of its vertices satisfy:
  - $d^+(v) = d^-(v)$ for all $v \neq v_1$ or v_2.
  - $d^+(v_1) = d^-(v_1)+1$
  - $d^-(v_2) = d^+(v_2)+1$

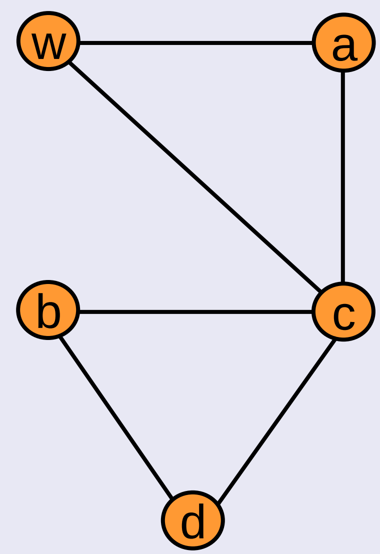
Finding Euler Tours

Fleury's Algorithm

- Applicable to undirected graphs
- Given a graph G, trace an euler tour
- CV : current vertex being visited
- E' : set of edges already traced
- EC : list of vertices in visiting order
- Start with vertex w

Fleury's Algorithm

```
EC = [w];
CV = w;
E' = {};
while |A(CV)| > 0 do
  if |A(CV)| > 1 then
    find a vertex v in A(CV) such that:
      (CV,v) is not a cut edge of G - E'
  else
    denote vertex in A(CV) by v;
  delete v in A(CV);
  delete CV in A(v);
  E' = E' ∪ {(CV,v)};
  CV = v;
  add CV to the tail of EC;
endwhile
```



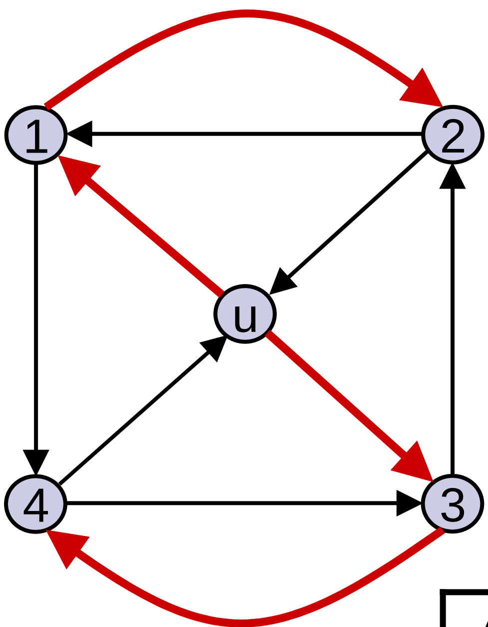
Finding Euler tour in digraph

- Construct an Euler tour starting with a spanning out-tree of the digraph.

Theorem: If G is connected, balanced digraph with a spanning out-tree T rooted at u, then an Euler tour can be traced in *reverse* direction as follows:

- The initial edge is any edge incident to u.
- Subsequent edges are chosen so as to be incident to the current vertex, such that:
  - no edge is traversed more than once
  - no edge of T is chosen if another edge is available
- The process stops when a vertex is reached with no unused edges incident to it.

Illustration



A_u	[2,4]
A_1	[u,2]
A_2	[1,3]
A_3	[u,4]
A_4	[1,3]

- Start with u
- Check A_u: 2 or 4
- Trace back to 2
- Check A_2: select 3
- Trace back to 3
- …

ET = u,3,4,u,1,2,1,4,3,2,u

The Chinese Postman Problem

- A postman picks up mail at the post office, delivers it, and returns to the post office.
  - He must cover each street in his area at least once.
  - He wishes to choose his route so that he walks as little as possible.
- First considered by a Chinese mathematician, Kuan (1962).

Representing the problem

- In a weighted graph, weight of a tour:

$$v_0 e_1 v_1 \ldots e_n v_0$$

$$\sum_{i=1}^{n} w(e_i)$$

- The problem is equivalent to find a minimum-weight tour (*optimal tour*) in a weighted connected graph with non-negative weights.
- If G is Eulerian, then any Euler tour is optimal.
  - An Euler tour traverses each edge only once.
  - Easily solved: Find an Euler tour.

Finding optimal tour

- If G is not Eulerian then any tour traverses some edges more than once.

- An edge e is said to be duplicated when its ends are joined by a new edge of weight $w(e)$.

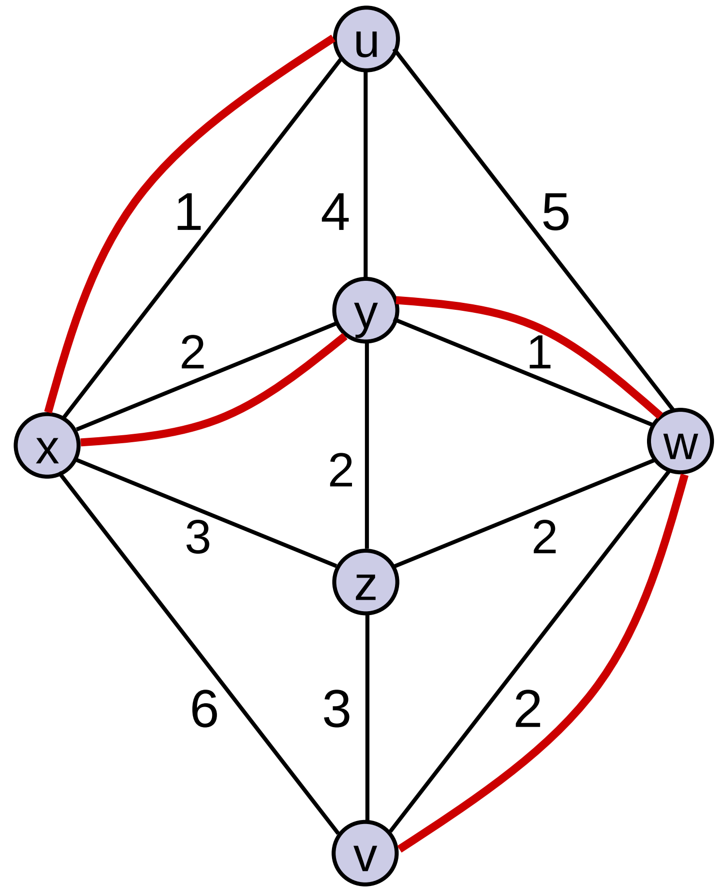
Lets rephrase the Chinese postman problem:

- Given a weighted graph G with non-negative weights:
  - Find an Eulerian weighted supergraph G* of G such that total weight of the new added edges is minimum.
  - Find an Euler tour in G*.

Finding the Eulerian supergraph

Special case:

- G has exactly two vertices of odd degree.
  - Assume these vertices are u and v.
- G* is obtained from G by duplicating each edge on a minimum-weight (u,v) path.



ET = xuywvzwyxuwvxzyx

General Solution

Problem: Find a shortest tour in a weighted, undirected, non-eulerian graph.

- Any vertex of odd-degree has at least one incident edge that is traversed at least twice.
- $r(u,v)$: number of times (u,v) is repeated
  - (u,v) is traversed $r(u,v) + 1$ times in the tour.
- The edge repetitions can be partitioned into a set of paths.
  - Each path has odd degree vertices as end-nodes.

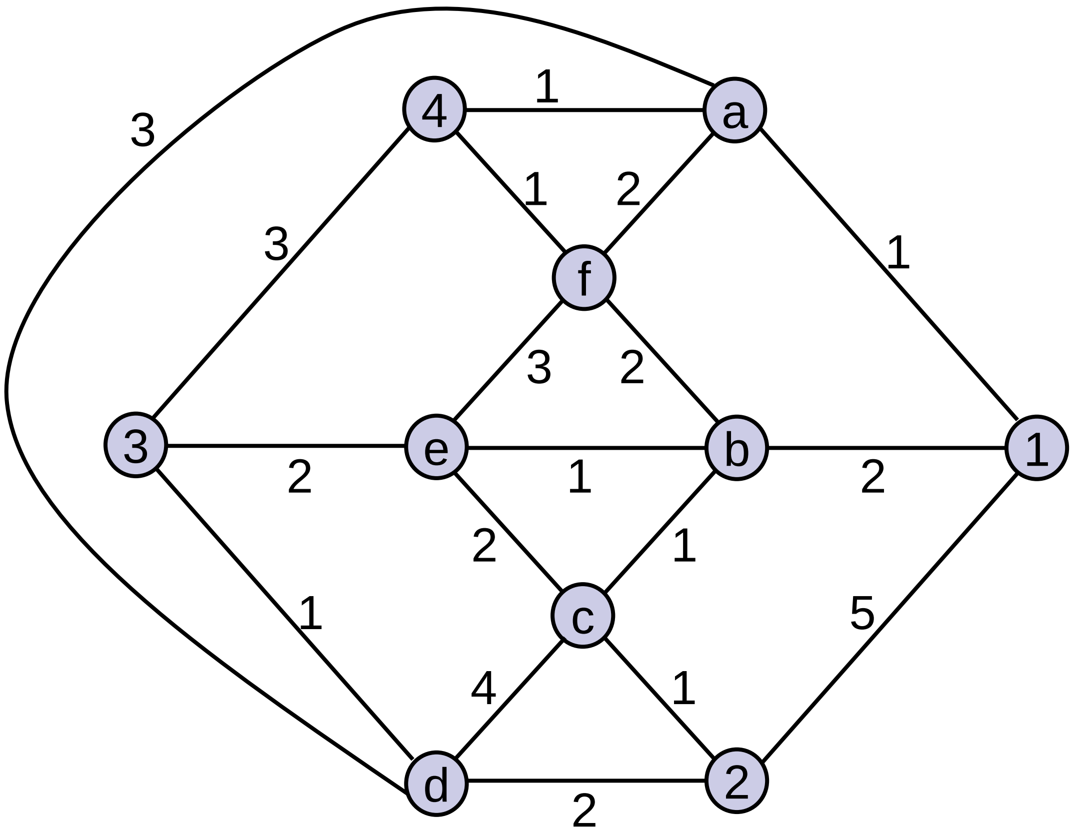
General solution

- Add to the original graph G, $r(u,v)$ repetitions of each edge (u,v)
  $\Rightarrow$resulting graph G'', is Eulerian.
- Postman’s problem becomes:
  Find a set of paths as described and such that sum of their edge weight is minimum.

Algorithm for undirected graphs

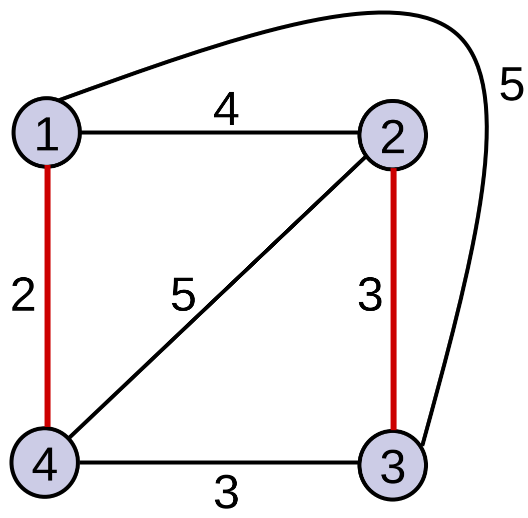
```
for all pairs of vertices of odd degree (u,v) do
  Find the shortest (u,v) path;
endfor;
Construct G' as follows:
  Vertex set of G' is the vertices of odd degree
  for each edge (u,v) do
    w(u,v) = distance(u,v) in G;
  endfor;
Find a minimum-weight perfect matching of G';
Construct G";
Find an Euler tour of G";
```

Example



d(1,2) = 4 : (1,b,c,2)
d(1,3) = 5 : (1,b,e,3)
d(1,4) = 2 : (1,a,4)
d(2,3) = 3 : (2,d,3)
d(2,4) = 5 : (2,c,b,f,4)
d(3,4) = 3 : (3,4)

Example



G'

Minimum-weight perfect matching:

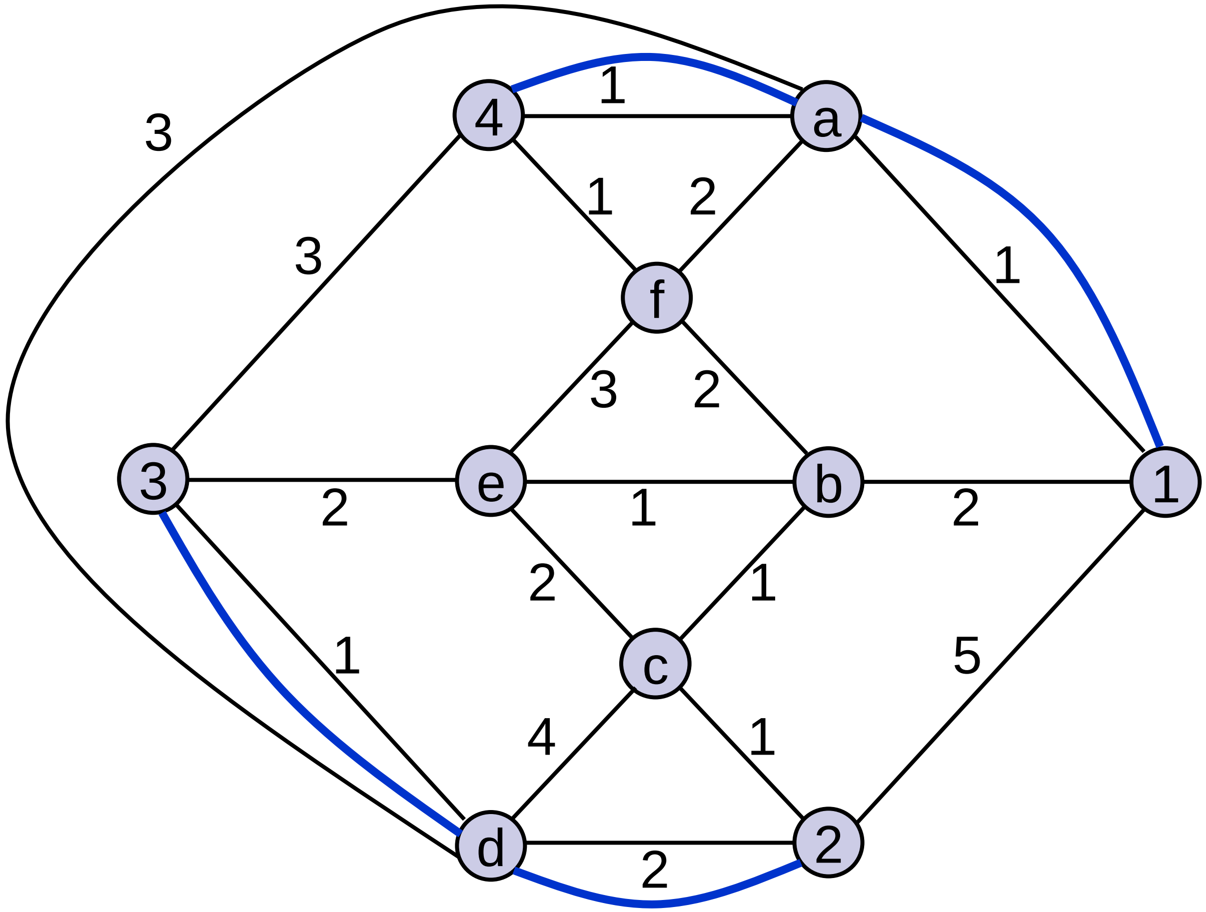
(1,4) and (2,3)

Duplicate edges along pahs:

(1,a,4)

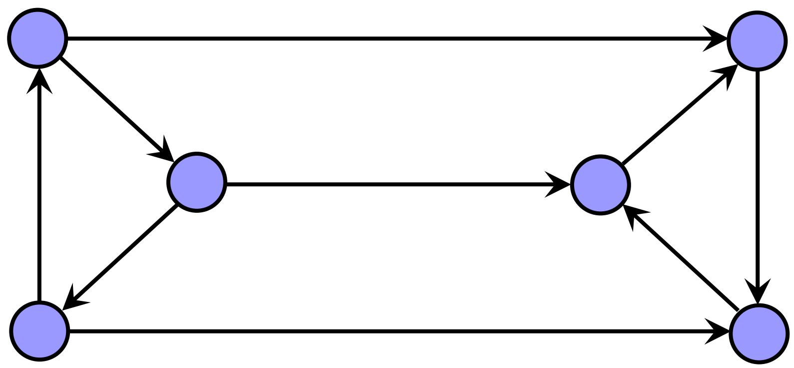
(2,d,3)

Example



Chinese Postman in digraphs

- Not all connected digraphs contain a solution.



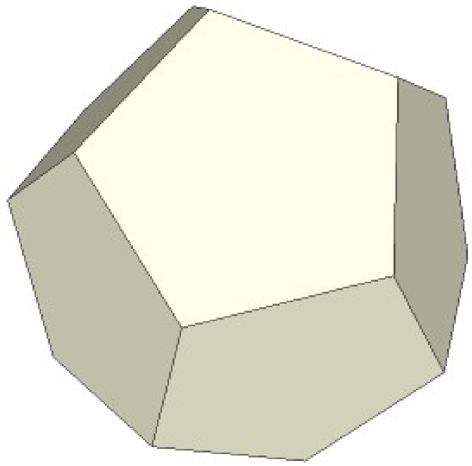
Theorem: A digraph has a Chinese postman's tour iff it is strongly connected.

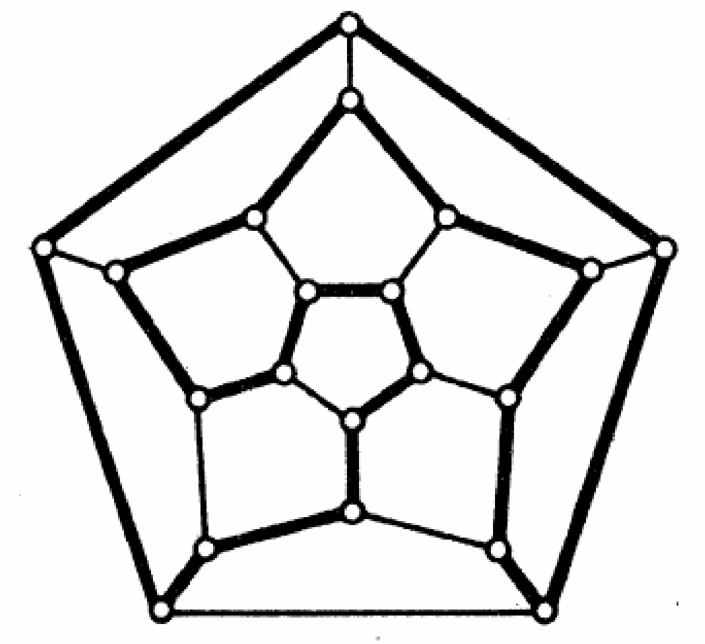
- Requires finding maximum flow, which we will study later.

Hamilton Cycle

Hamilton path: A path that contains every vertex of G.

Hamilton cycle: A cycle that contains every vertex of G.

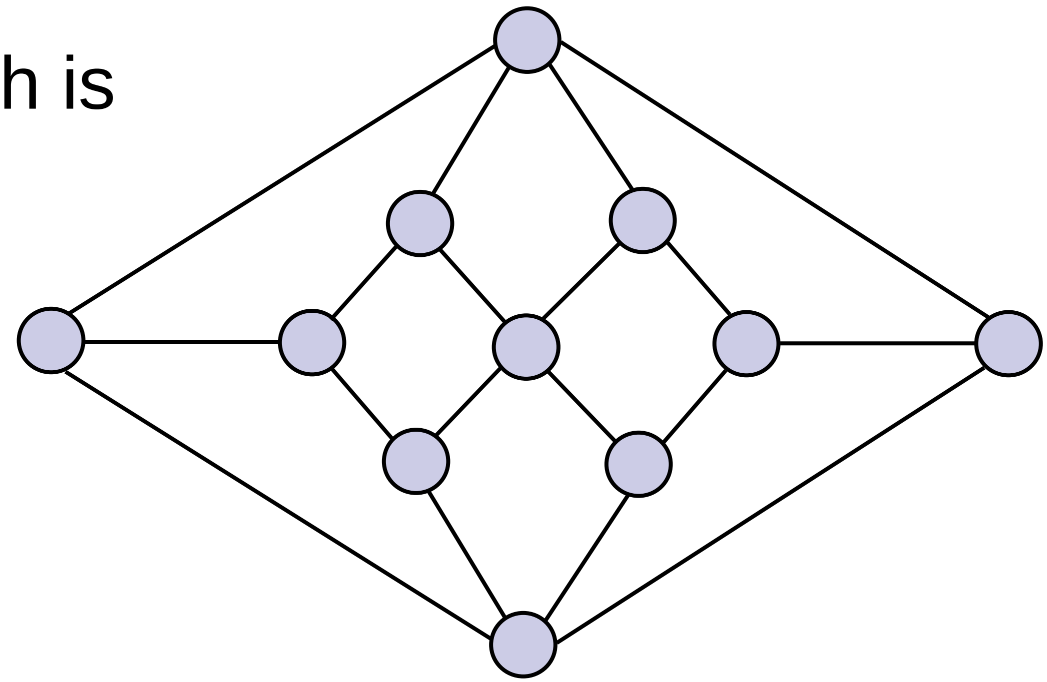




- Named after Hamilton.
- A game on dodecahedron.
- The dodecahedron is hamiltonian.

Hamilton Cycle

- The Herchel graph is nonhamiltonian.

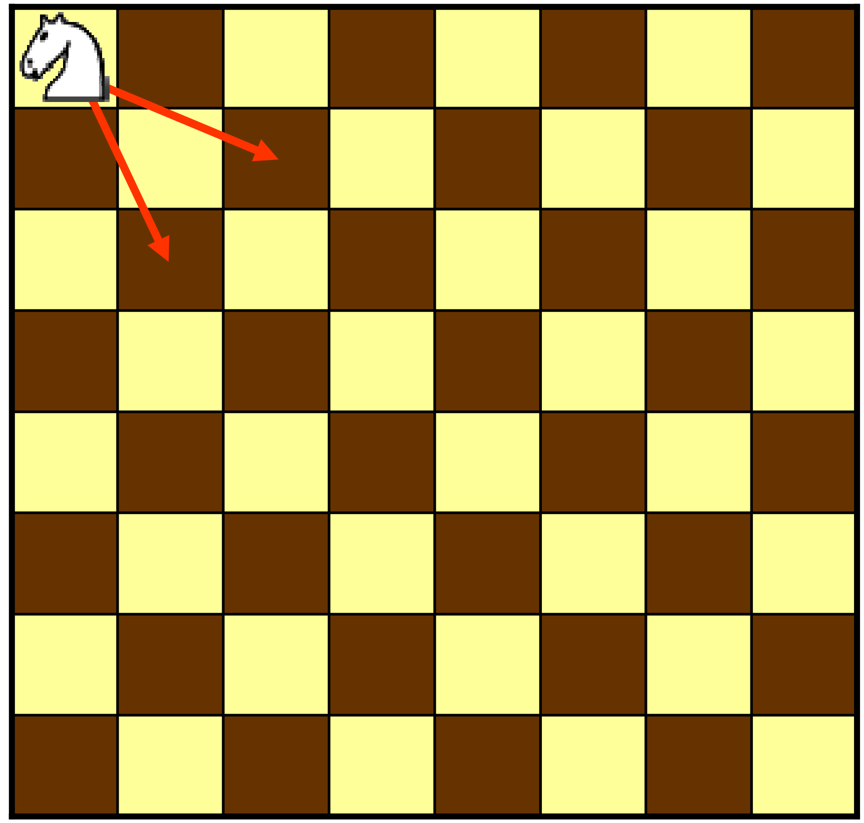


- No necessary and sufficient condition for a graph to be hamiltonian is known.
- One of the main unsolved problems of graph theory.

Knight’s Tour

- Puzzles and board games often involve Hamilton cycles.
- Knight’s tour of a chessboard:

  A sequence of knight’s moves which:
  - visit every square of a chessboard precisely once,
  - and returns to its initial square.



How do you represent this problem as a Hamilton cycle?

Theorems on Hamilton cycles

- There are several theorems that provide some useful necessary or sufficient conditions.

Theorem h.1: If G is hamiltonian then for every nonempty proper subset S of V:

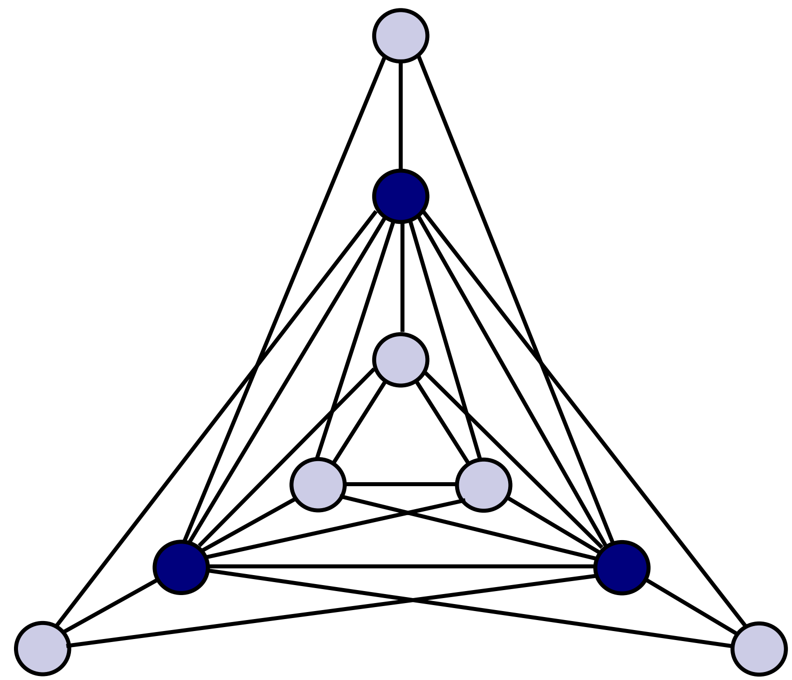
$$\omega(G-S) \leq |S|$$

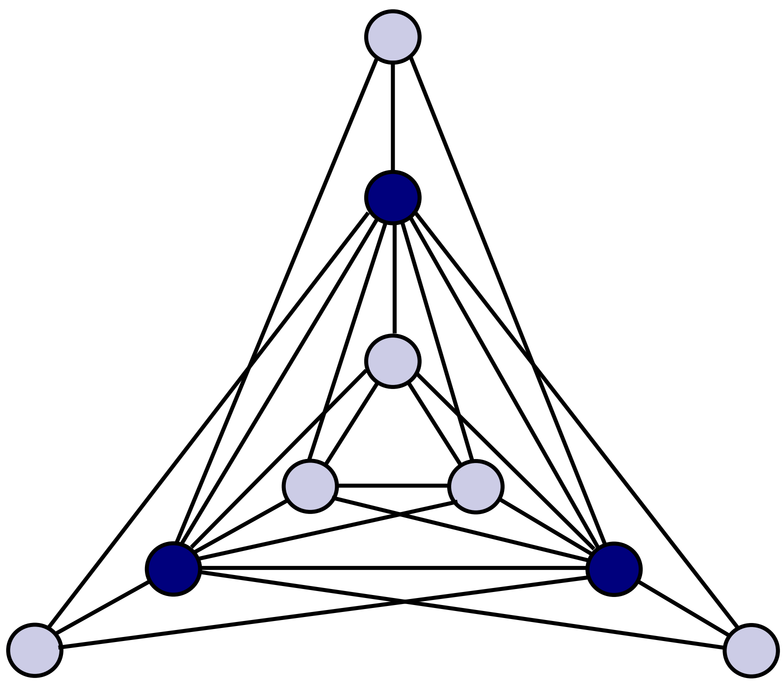
ω: number of components

This theorem can sometimes be applied to show that a particular graph is nonhamiltonian.

Example

- 9 vertices
- Delete 3 dark colored vertices
  $\Rightarrow$ 4 components remain.





4 > 3
$\Rightarrow$ This graph is nonhamiltonian.

Sufficient conditions

Dirac's condition

Theorem h.2: If G is a simple graph with:

- $|V| \geq 3$
- $\delta \geq |V|/2$

then G is hamiltonian.

Bondy and Chvatal

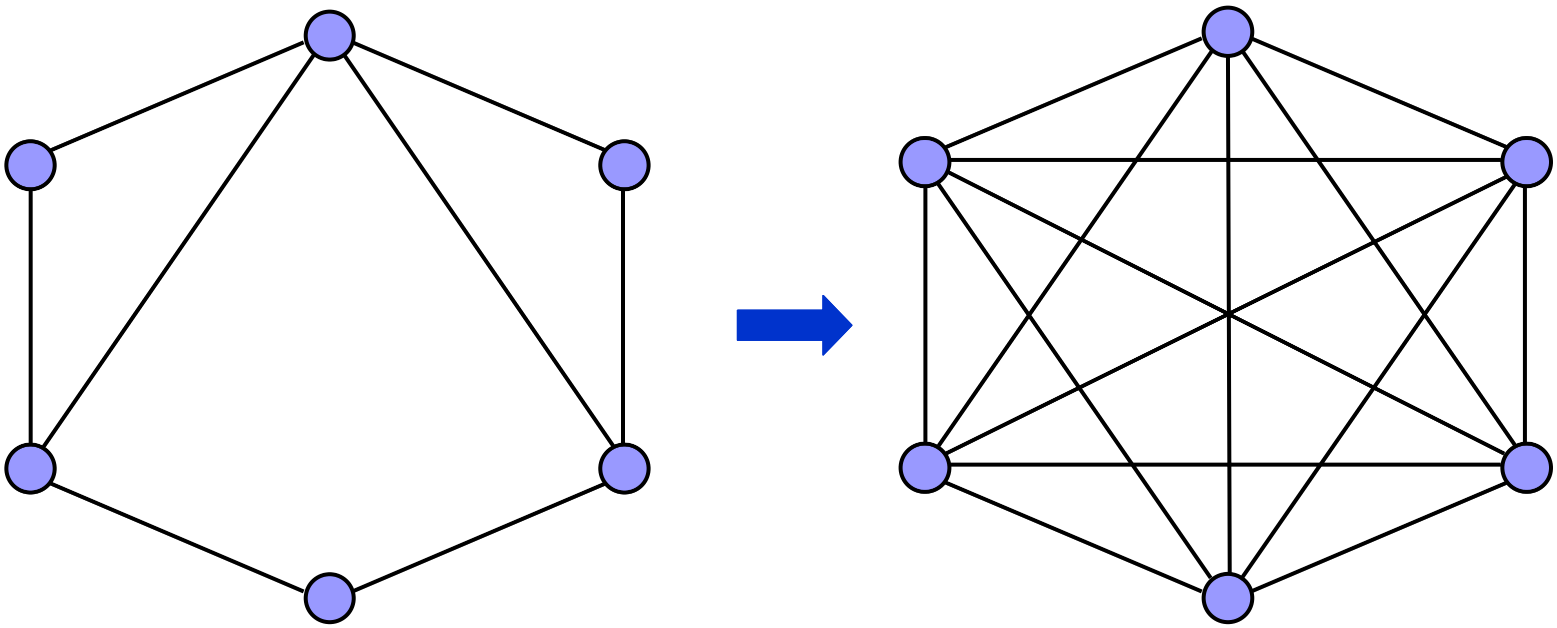
Lemma h.2.1: If G is a simple and u and v are nonadjacent vertices of G such that:

$$d(u) + d(v) \geq |V|$$

then G is hamiltonian iff G + (u,v) is hamiltonian.

Closure

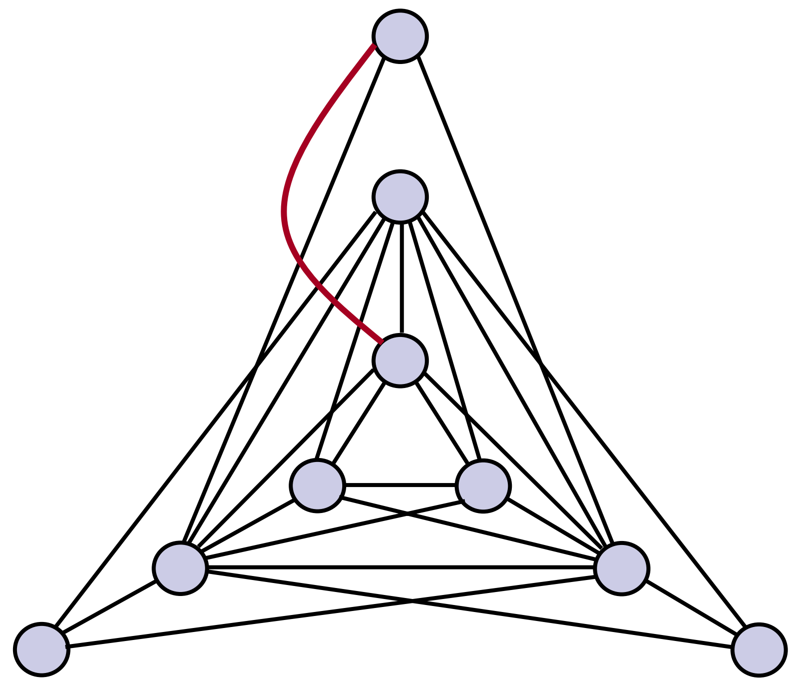
- The closure of G, c(G) is the graph obtained from G by recursively joining *pairs of nonadjacent vertices whose degree sum is at least |V|,* until no such pair remains.



More theorems…

Theorem h.3: A simple graph is hamiltonian iff its closure is hamiltonian.

Corollary h.3: Let G be a simple graph with $|V| \geq 3$. If c(G) is complete then G is hamiltonian.



- The closure of the above graph is complete.
- By corollary h.3 this graph is hamiltonian.

Hamilton paths on digraphs

Theorem h.4: A digraph whose underlying graph is complete, contains a Hamilton path.

Theorem h.5: A strongly connected digraph whose underlying graph is complete is Hamiltonian.

A more general sufficient condition

Theorem h.6: Let G be a simple graph with degree sequence $(d_1, d_2, \ldots, d_n)$, where:

- $d_1 \leq d_2 \leq \ldots \leq d_n$
- $n \geq 3$

Suppose that there is no value of m less than $n/2$ for which:

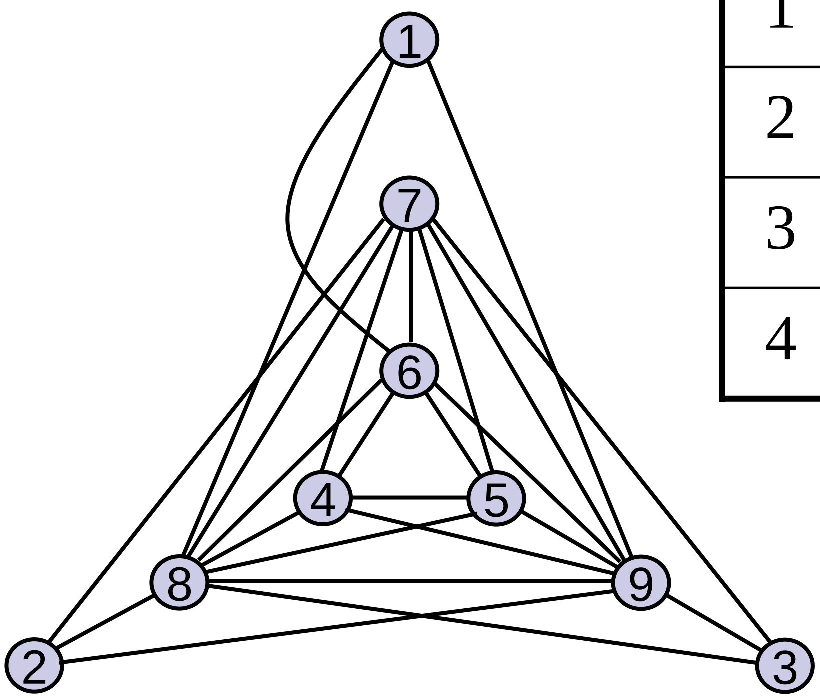
- $d_m \leq m$ and
- $d_{n-m} < n - m$

Then G is hamiltonian.

Example

- Degree sequence: (3,3,3,5,5,6,7,8,8)
- $1 \leq m < 4.5$

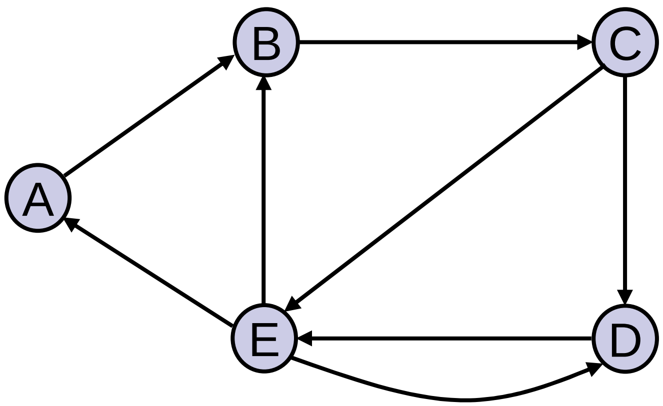
m	$d_m \leq m$		$d_{n-m} < n - m$	
1	$d_1 \leq 1$	No		
2	$d_2 \leq 2$	No		
3	$d_3 \leq 3$	Yes	$d_6 < 6$	No
4	$d_4 \leq 4$	No		



Finding all Hamilton cycles

- A straightforward technique to generate all the Hamilton cycles (paths) of a graph or digraph.
- Inefficient algorithm
- We will use matricial products.
- Start with adjacency matrix, and obtain M_1 by:
  - replacing any (i,j)-th non-zero entry with string ij.
  - replacing any non-zero diagonal by 0.
- Define a second matrix M, derived from M_1 by deleting the initial letter in each element.

Illustration



$M_1 =$

0	AB	0	0	0
0	0	BC	0	0
0	0	0	CD	CE
0	0	0	0	DE
EA	EB	0	ED	0

$M =$

0	B	0	0	0
0	0	C	0	0
0	0	0	D	E
0	0	0	0	E
A	B	0	D	0

Finding all Hamilton cycles

- Define a marticial product from which we can generate M_j for all $1 < j <$ n.

$$M_j = M_{j-1} * M$$

where the (r,s)-th element of M_j is defined as follows:

$$M_j(r,s) = \left\{ M_{j-1}(r,t)\, M(t,s) \right\}$$

$$1 \le t \le n$$

neither $M_{j-1}(r,t)$ nor $M(t,s)$ are zero or have a common vertex.

Illustration

$M_1 =$

0	AB	0	0	0
0	0	BC	0	0
0	0	0	CD	CE
0	0	0	0	DE
EA	EB	0	ED	0

$M_2 =$

0	0	ABC	0	0
0	0	0	BCD	BCE
CEA	CEB	0	CED	CDE
DEA	DEB	0	0	0
0	EAB	EBC	0	0

$M =$

0	B	0	0	0
0	0	C	0	0
0	0	0	D	E
0	0	0	0	E
A	B	0	D	0

$M_3 =$

0	0	0	ABCD	ABCE
BCEA	0	0	BCED	BCDE
CDEA	CEAB CDEB	0	0	0
0	DEAB	DEBC	0	0
0	0	EABC	EBCD	0

Illustration

$M_4 =$

0	0	0	ABCD	ABCE
BCEA	0	0	BCED	BCDE
CDEA	CEAB CDEB	0	0	0
0	DEAB	DEBC	0	0
0	0	EABC	EBCD	0

- Each element is a set of paths.
- M_4 displays all Hamilton paths of the example graph.
- By checking the endpoints of the paths, we obtain a single Hamilton cycle: ABCDEA

The Travelling Salesman Problem

- A salesman wishes to:
  - visit a number of towns, and then
  - return to his starting town.
- Given the travelling times between towns, how should the travel be planned, so that:
  - he visits each town exactly once, and
  - he travels in as short time as possible.
- This is equivalent to find a minimum-weight Hamilton cycle in a weighted complete graph.

The Travelling Salesman Problem

- No efficient algorithm to solve TSP is known.
- It is desirable to have a method to obtain a reasonably good solution.
- A simple approach:
  - Find a Hamilton cycle C,
  - Search for another of smaller weight by modifying C:

Let C = $v_1v_2\ldots v_nv_1$

For all i and j such that $1 < i + 1 < j < n$, we can obtain a new Hamilton cycle:

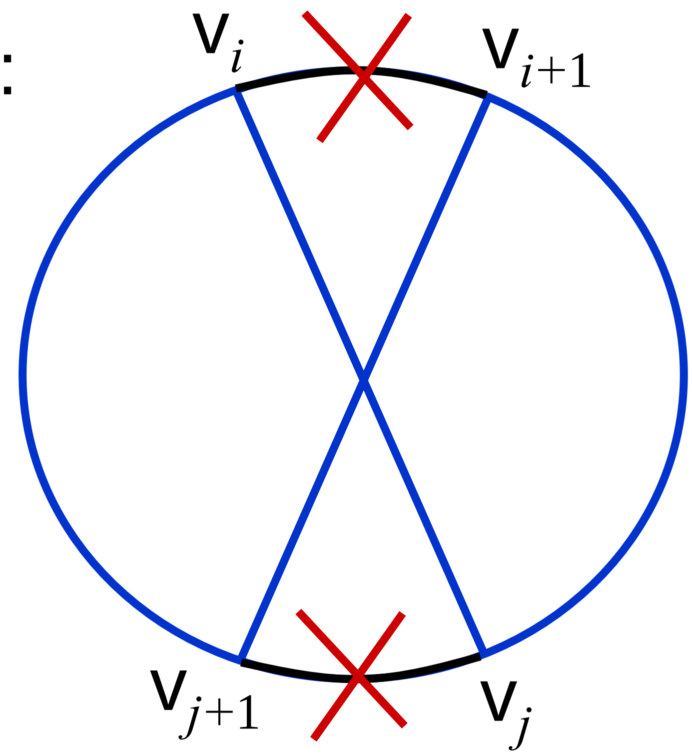
$$C_{ij} = v_1v_2\ldots v_iv_jv_{j-1}\ldots v_{i+1}v_{j+1}v_{j+2}\ldots v_nv_1$$

A simple approach

- This new cycle is obtained by:
- deleting edges $v_i v_{i+1}$ and $v_j v_{j+1}$
- and adding edges $v_i v_j$ and $v_{i+1} v_{j+1}$
- If for some i and j,

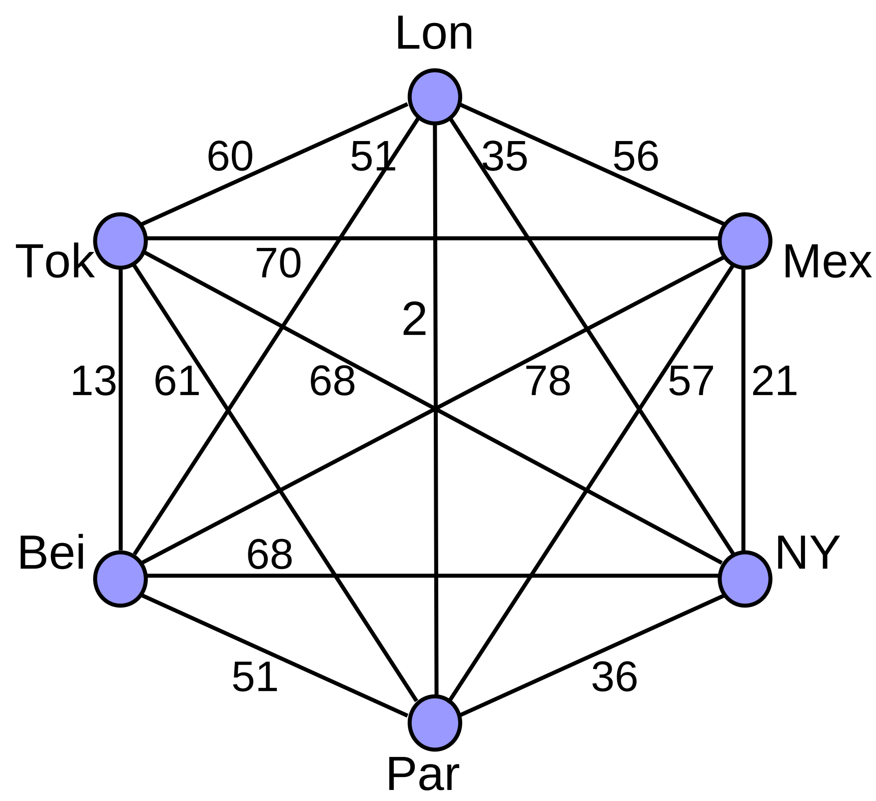
$$w(v_i v_j) + w(v_{i+1} v_{j+1}) < w(v_i v_{i+1}) + w(v_j v_{j+1})$$

C_{ij} is an improvement on C.

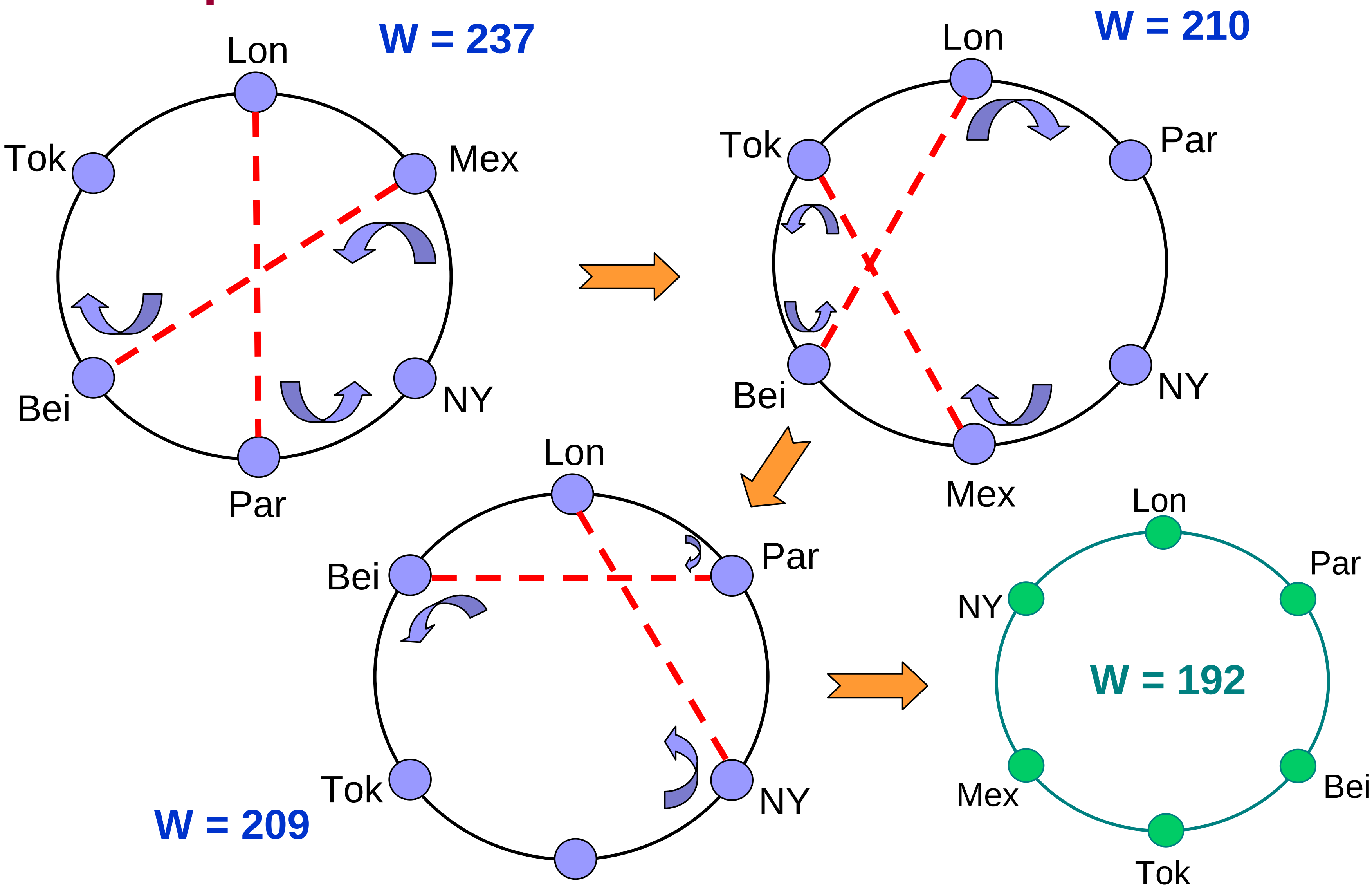


A simple approach

- The modification can be repeated in sequence, until the cycle cannot be improved further.
- The procedure can be repeated several times, starting with a different cycle each time.



Example

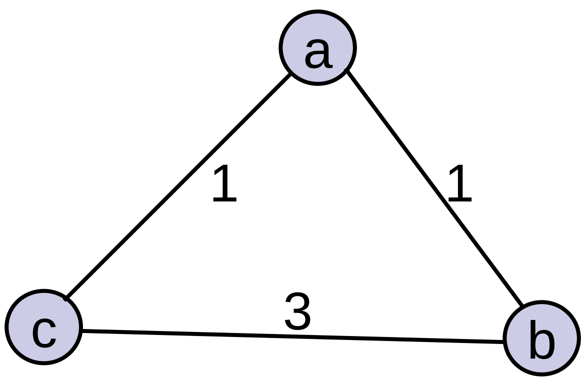


W = 237
Lon
Tok
Mex
Bei
NY
Par
W = 210
Lon
Tok
Par
Bei
NY
Mex
Lon
Bei
Par
Tok
NY
Mex
W = 209
Lon
NY
Par
W = 192
Mex
Bei
Tok

TSP– A variation

- Find a minimum-weight cycle which visits every vertex *at least once*.
- A solution to this problem is not necessarily a simple cycle.

Example:



Solution: abaca

Triangle inequality

- If for every pair of vertices u and v of a graph G, the weights satisfy:

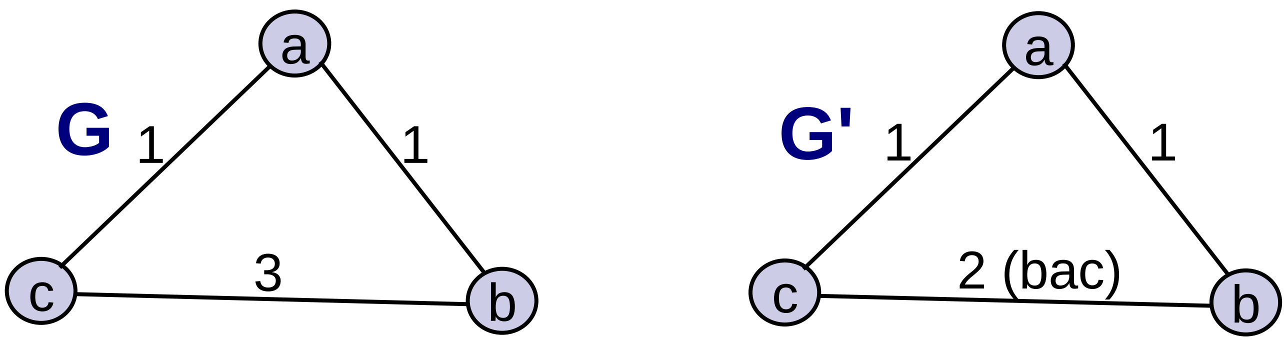
  $$w(\mathrm{u},\mathrm{v}) \leq w(\mathrm{u},\mathrm{x}) + w(\mathrm{x},\mathrm{v})$$

  for all vertices $\mathrm{x} \neq \mathrm{u},\mathrm{v}$,

  then the triangle inequality is satisfied in G.
- If the triangle inequality does not hold in a graph, then it is likely that the second variation of TSP is not a simple cycle.
- There is a technique to transform the TSP for any graph G, into the problem of finding Hamilton cycle in another graph G'.

Transforming graphs

- G' is a complete graph with:
  - V' = V
  - Each edge (u,v) in E' has a weight equal to minimum distance of (u,v).
  - Each edge of G' corresponds to a path of one or more edges of G.



Theorem: A solution to TSP in G corrsponds to, and is of the same weight as a minimum-weight Hamilton cycle in the complete graph G'.

Solving TSP

- For a complete undirected graph with n vertices, there are $(n-1)!\,/\,2$ different Hamilton cycles.
- The number of addition operations required to find the lengths of all these cycles is $O(n!)$.
- Given a computer that can perform these additions at a rate of 10^9/second, the computation times are as follows:

n	$\sim n!$	Time
12	4.8×10^8	0.5 sec
15	1.3×10^{12}	18 min
20	2.4×10^{18}	80 years
50	3.0×10^{64}	10^{48} years

Approximation algorithms

- It is useful to have a polynomial-time algorithm which produce, within known bounds, an approximation to the required result.
- Let:
  - L : the value obtained by an approximation algorithm.
  - L_0: the exact value of the solution.
- We require a performance guarantee in form:

$$1 \leq L / L_0 \leq \alpha$$

  - We would like α to be as close to 1 as possible.

Nearest neighbor method

- Start at vertex v_1
- Trace (v_1,v_2) which is the shortest edge from v_1.
- Leave v_2 along (v_2,v_3) the shortest edge from v_2.
  - Keep the cycle simple.
- Continue until every vertex has been visited.
- Complete the cycle by edge (v_n,v_1).
- It can be shown that, for this algorithm:

$$\alpha = \frac{1}{2}(\lceil \ln n \rceil + 1)$$

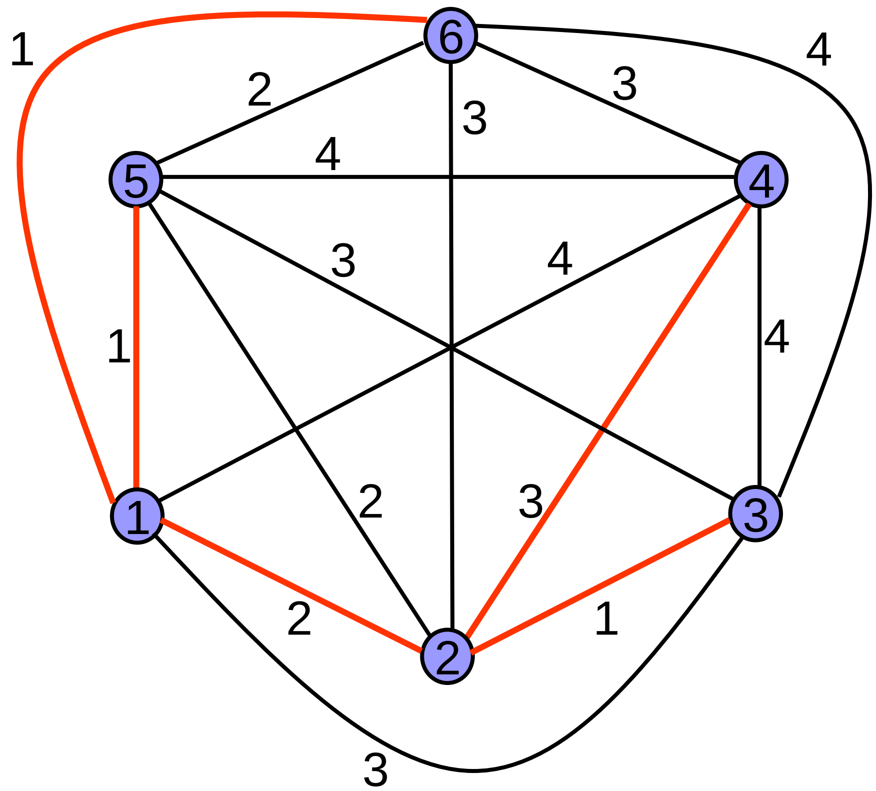
Twice-around-the-MST algorithm

```
1.Find a minimum-weight spanning tree T of G;
2.Conduct a DFS of T:
    associate a DFS index L(v) with each vertex;
3.Output the following cycle:
    C = vi1,vi2,…,vin,vi1
  where
    L(vij) = j
```

- Hamilton cycle visits the vertices in the order of their depth-first indices.

Theorem: The twice-around-the-MST algorithm gives $\alpha < 2$.

Illustration



DFS =
1,2,3,2,4,2,1,5,1,6,1

C = 1,2,3,4,5,6,1