Introduction to Graph Theory

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 351

Subsection 1

Connectivity

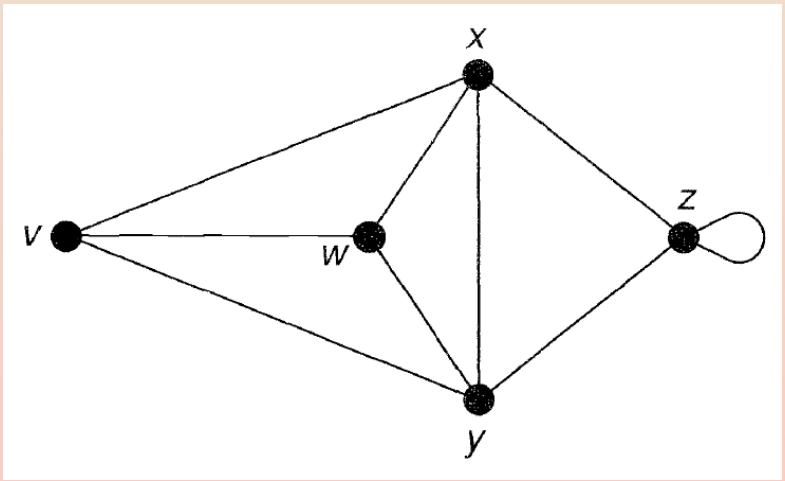
Walks

- Given a graph G, a **walk** in G is a finite sequence of edges of the form $v_0v_1, v_1v_2, \ldots, v_{m-1}v_m$, also denoted by $v_0 \to v_1 \to v_2 \to \cdots \to v_m$, in which any two consecutive edges are adjacent or identical.
- Such a walk determines a sequence of vertices $v_0, v_1, \ldots, v_m$.

  We call v_0 the **initial vertex** and v_m the **final vertex** of the walk, and speak of a **walk from v_0 to v_m**.
- The number of edges in a walk is called its **length**.

  Example:

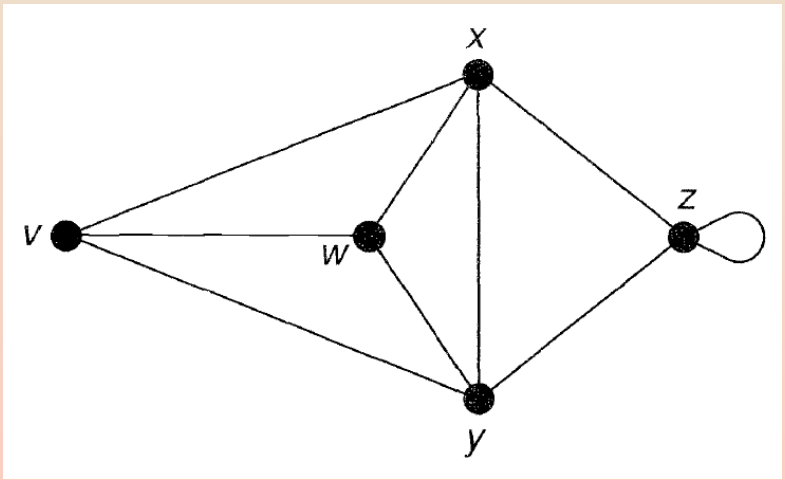
  In the following graph, $v \to w \to x \to y \to z \to z \to y \to w$ is a walk of length 7 from v to w.



Trails and Paths

- A walk in which all the edges are distinct is a **trail**.
- If, in addition, the vertices $v_0, v_1, \ldots, v_m$ are distinct (except, possibly, $v_0 = v_m$), then the trail is a **path**.
- A path or trail is **closed** if $v_0 = v_m$, and a closed path containing at least one edge is a **cycle**.
- Note that any loop or pair of multiple edges is a cycle.

  Example:



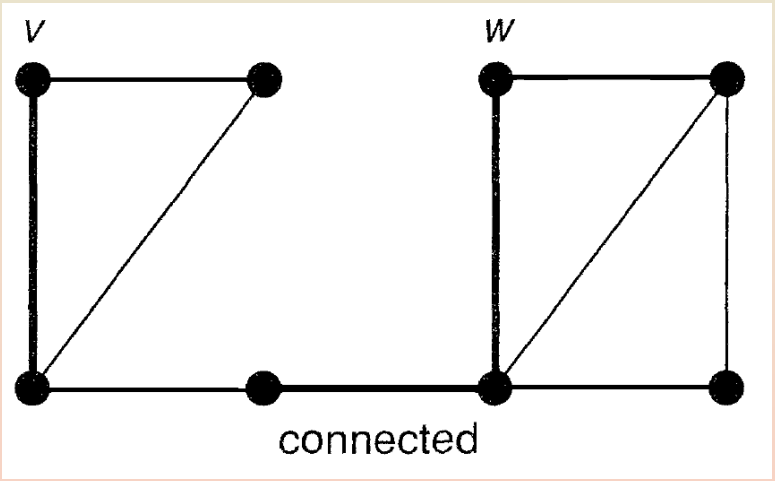
$v \to w \to x \to y \to z \to z \to x$ is a trail.
$v \to w \to x \to y \to z$ is a path.
$v \to w \to x \to y \to z \to x \to v$ is a closed trail.
$v \to w \to x \to y \to v$ is a cycle.

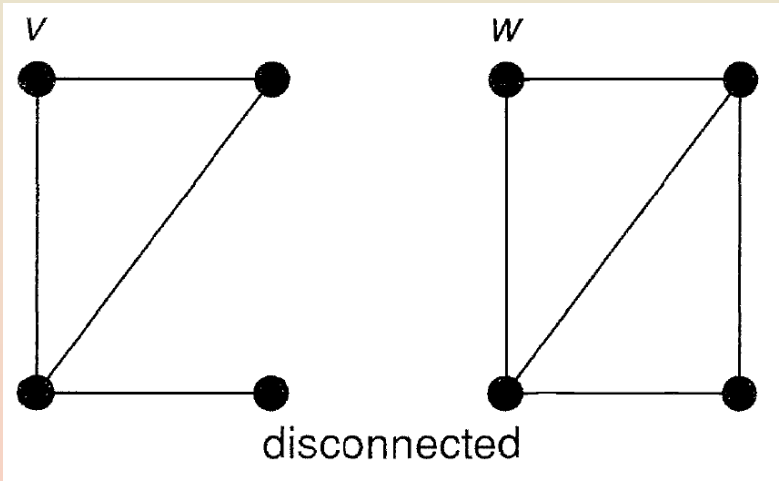
- A cycle of length 3, such as $v \to w \to x \to v$, is a **triangle**.

Connected Graphs

- A graph is **connected** if and only if there is a path between each pair of vertices.

  Example:





Bipartite Graphs and Cycles

- Note also that G is a bipartite graph if and only if each cycle of G has even length.
- We prove only one half of this result, leaving the proof of its converse as an exercise:

Theorem

If G is a bipartite graph, then each cycle of G has even length.

- Since G is bipartite, we can split its vertex set into two disjoint sets A and B so that each edge of G joins a vertex of A and a vertex of B.

  Let $v_0 \to v_1 \to \cdots \to v_m \to v_0$ be a cycle in G, and assume (without loss of generality) that v_0 is in A. Then v_1 is in B, v_2 is in A, and so on. Since v_m must be in B, the cycle has even length.

Bounding the Number of Edges

- Consider the number of edges of a simple connected graph on n vertices:
  - The graph has fewest edges when it has no cycles;
  - The graph has most edges when it is a complete graph.

  Thus, the number of edges must lie between $n-1$ and $\frac{n(n-1)}{2}$.

Theorem

Let G be a simple graph on n vertices. If G has k components, then the number m of edges of G satisfies $n-k \leq m \leq \frac{(n-k)(n-k+1)}{2}$.

- We prove $m \geq n-k$ by induction on the number of edges of G:
  - The result is trivial if G is a null graph.
  - If G contains as few edges as possible (say m_0), then the removal of any edge of G must increase the number of components by 1. The graph that remains has n vertices, $k+1$ components, and m_0-1 edges. By the induction hypothesis, $m_0 - 1 \geq n-(k+1)$, giving $m_0 \geq n-k$, as required.

Proving the Upper Bound

- To prove the upper bound, we can assume that each component of G is a complete graph. Suppose, then, that there are two components C_i and C_j with n_i and n_j vertices, respectively, where $n_i \geq n_j > 1$.

  If we replace C_i and C_j by complete graphs on $n_i + 1$ and $n_j - 1$ vertices, then the total number of vertices remains unchanged, and the number of edges is changed by

$$\frac{(n_i+1)n_i - n_i(n_i-1)}{2} - \frac{n_j(n_j-1) - (n_j-1)(n_j-2)}{2} = n_i - n_j + 1 > 0.$$

  It follows that, in order to attain the maximum number of edges, G must consist of a complete graph on $n - k + 1$ vertices and $k - 1$ isolated vertices. The result now follows.

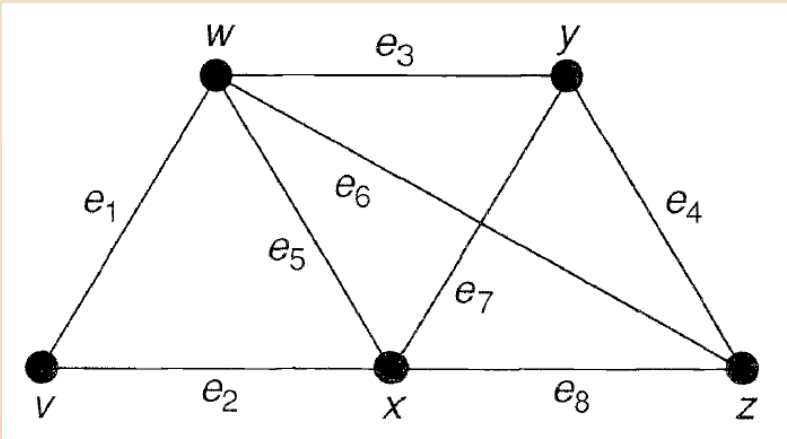
Corollary

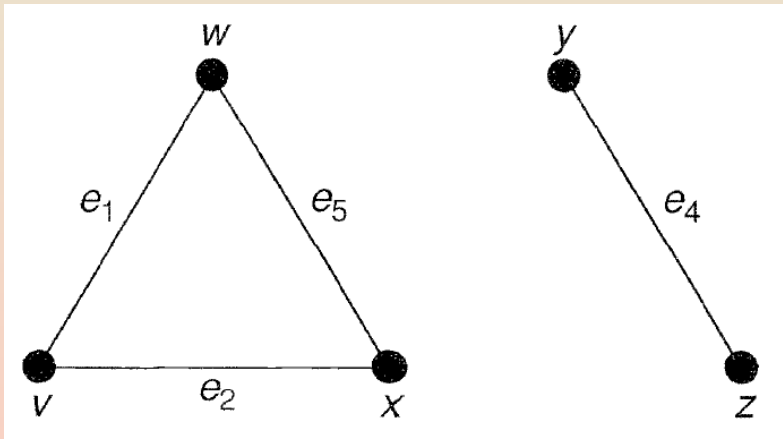
Any simple graph with n vertices and more than $\frac{(n-1)(n-2)}{2}$ edges is connected.

Disconnecting Sets

- We ask "how connected is a connected graph?"
- One interpretation of this is to ask how many edges or vertices must be removed in order to disconnect the graph.
- A **disconnecting set** in a connected graph G is a set of edges whose removal disconnects G.

  Example: In the graph of the figure on the left the sets

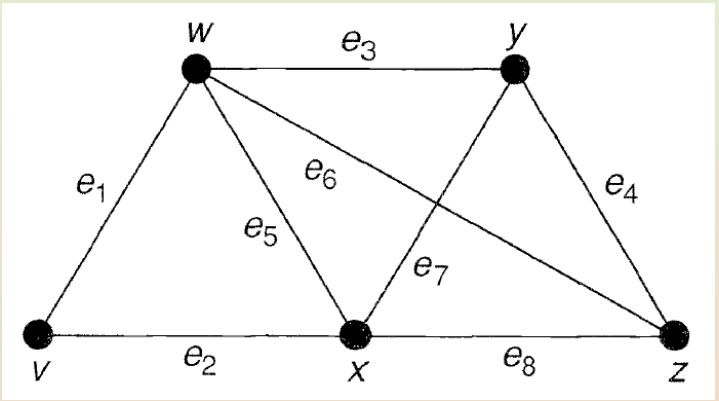


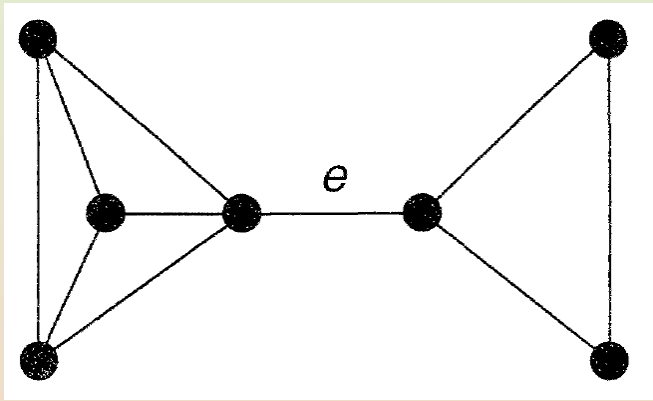


$\{e_1, e_2, e_5\}$ and $\{e_3, e_6, e_7, e_8\}$ are disconnecting sets of G. The disconnected graph left after removal of the second is on the right.

Cutsets and Bridges

- We define a **cutset** to be a disconnecting set, no proper subset of which is a disconnecting set.
  Example: $\{e_3, e_6, e_7, e_8\}$ is a cutset.



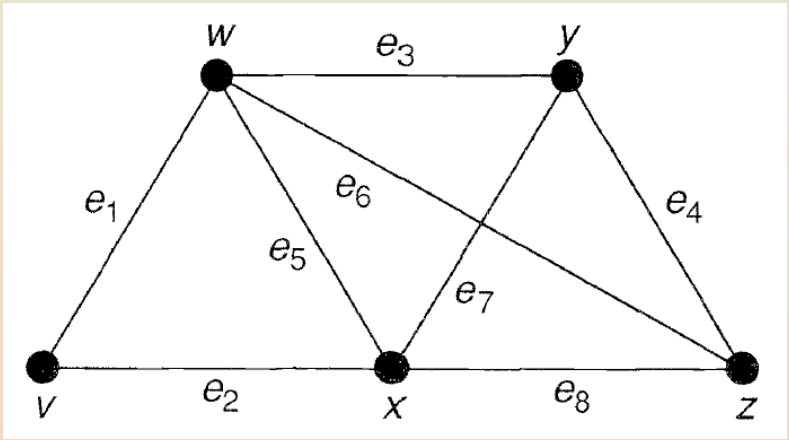


- The removal of the edges in a cutset always leaves a graph with exactly two components.
- If a cutset has only one edge e, we call e a **bridge** (right figure).
- For (possibly disconnected) graphs G,
  - a **disconnecting set** of G is a set of edges whose removal increases the number of components of G;
  - a **cutset** of G is a disconnecting set, no proper subset of which is a disconnecting set.

Edge Connectivity

- If G is connected, its **edge connectivity** $\lambda(G)$ is the size of the smallest cutset in G.
- Thus $\lambda(G)$ is the minimum number of edges that we need to delete in order to disconnect G.

  Example:

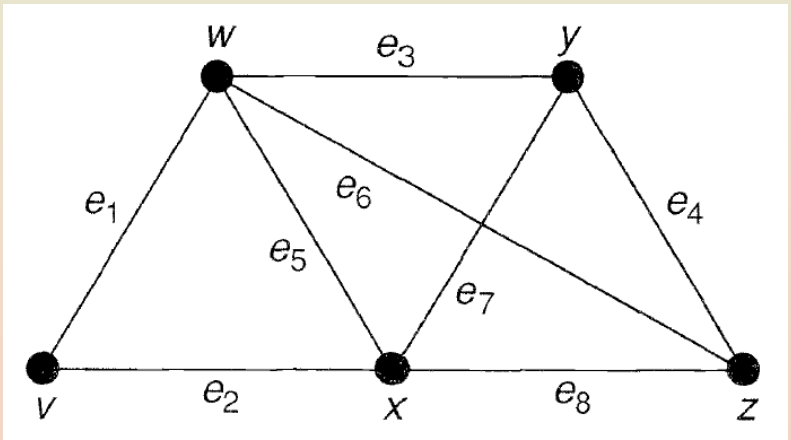


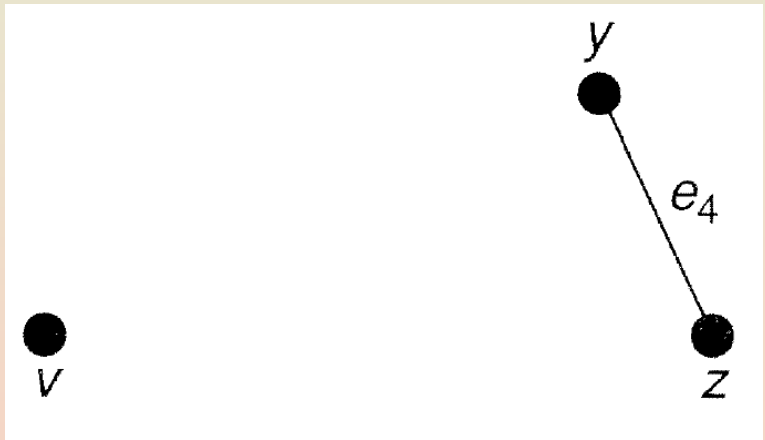
  For G in the figure, $\lambda(G) = 2$, corresponding to the cutset $\{e_1, e_2\}$.
- We also say that G is k-**edge connected** if $\lambda(G) \geq k$.

  Example: The graph above is 1-edge connected and 2-edge connected, but not 3-edge connected.

Separating Sets and Cut-Vertices

- We also need the analogous concepts for the removal of vertices:
- A **separating set** in a connected graph G is a set of vertices whose deletion disconnects G or leaves only one vertex; recall that when we delete a vertex, we also remove its incident edges.
  Example:



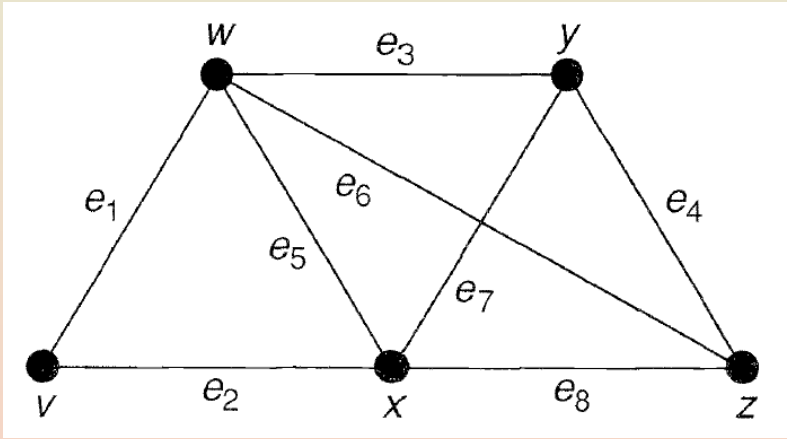


  In the graph of the figure, the sets $\{w, x\}$ and $\{w, x, y\}$ are separating sets of G; the disconnected graph left after removal of the first is shown on the right.
- If a separating set contains only one vertex v, we call v a **cut-vertex**.

Connectivity

- If G is connected and not a complete graph, its (**vertex**) **connectivity** $\kappa(G)$ is the size of the smallest separating set in G.
- Thus $\kappa(G)$ is the minimum number of vertices that we need to delete in order to disconnect G.

  Example: If G is the graph in the figure, then $\kappa(G) = 2$, corresponding to the separating set $\{w, x\}$.



- We also say that G is k-**connected** if $\kappa(G) \geq k$.

  Example: The graph above is 1-connected and 2-connected, but not 3-connected.

Connectivity and Edge-Connectivity

Lemma

For any graph G with n vertices $\kappa(G) \leq n-1$.

- Clearly, removing $n-1$ vertices leaves only one vertex.
  Thus, we need to remove at most $n-1$ vertices to disconnect the graph. It follows that $\kappa(G) \leq n-1$.

Theorem

For any graph G with n vertices,

$$\kappa(G) \leq \lambda(G).$$

- Consider a smallest cut set $[S, S']$ separating the graph into two sets of vertices S and S'.
  Note that, if S has m vertices, then $n \geq m+1$.
  So $(m-1)n \geq (m-1)(m+1) \Rightarrow mn - n \geq m^2 - 1$
  $\Rightarrow m(n-m) \geq n-1$ or $|S| \cdot |S'| \geq n-1$.

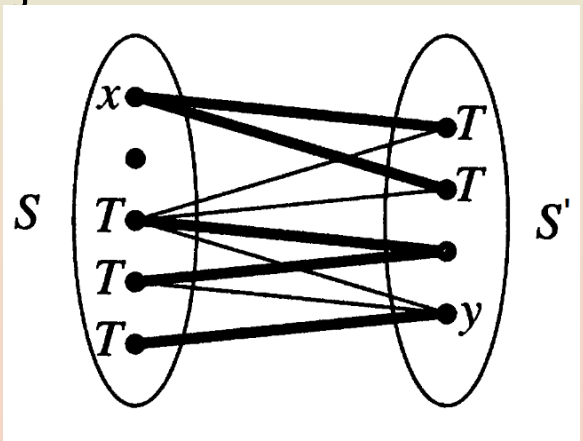
Connectivity and Edge-Connectivity (Cont'd)

- We consider two cases:
  - If every vertex of S is adjacent to every vertex in S', then

$$\lambda(G) = |[S, S']| = |S| \cdot |S'| \geq n - 1 \geq \kappa(G).$$

  - Otherwise, choose $x \in S$ and $y \in S'$ not adjacent.

    Let T be the set of neighbors of x in S' together with those vertices in $S - \{x\}$ with neighbors in S'.
    Note that $|T| \leq |[S, S']|$.



    Every path from x to y passes through a vertex in T.
    So T is a separating set.
    Now we have

$$\lambda(G) = |[S, S']| \geq |T| \geq \kappa(G).$$

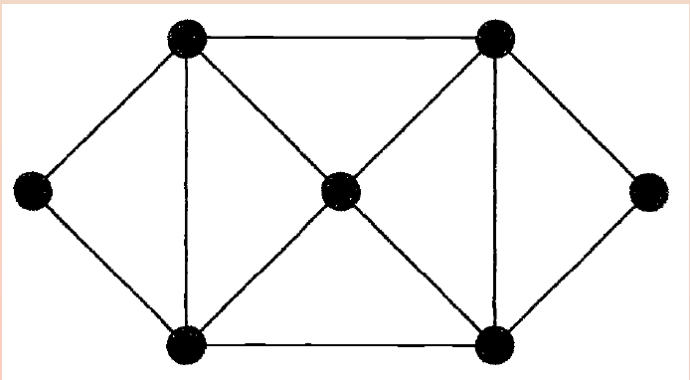
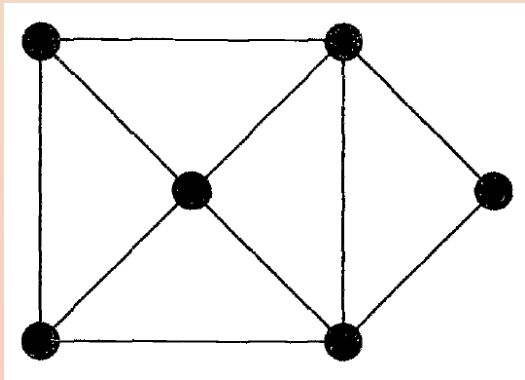
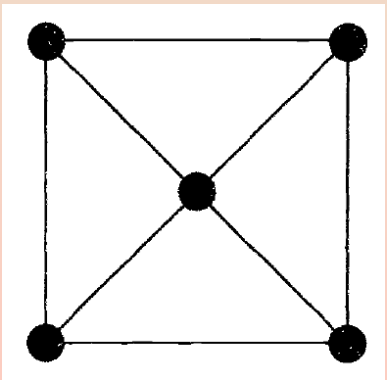
Subsection 2

Eulerian Graphs

Eulerian and Semi-Eulerian Graphs

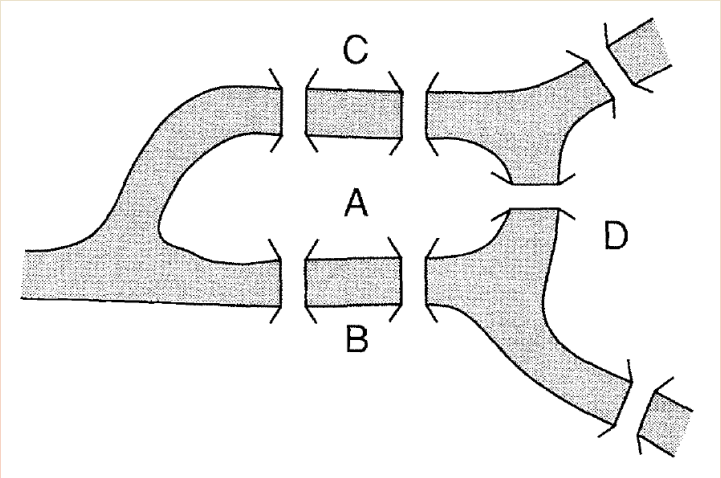
- A connected graph G is **Eulerian** if there exists a closed trail containing every edge of G.
- Such a trail is an **Eulerian trail**.
- The definition requires each edge to be traversed once and once only.
- A *non-Eulerian* graph G is **semi-Eulerian** if there exists a trail containing every edge of G.

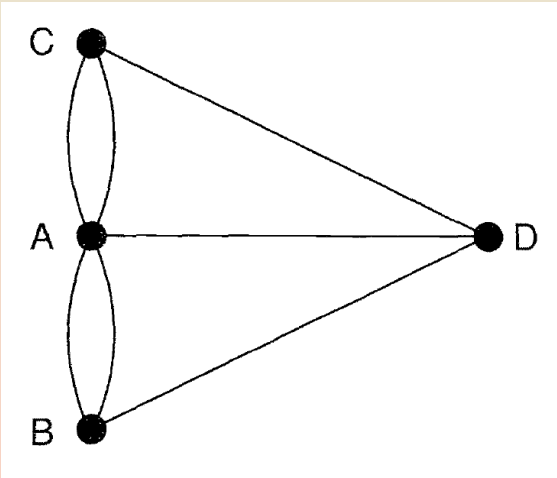
  Example: An Eulerian, semi-Eulerian and non-Eulerian graph, respectively.

The Königsberg Bridges Problem

- A typical problem related to Eulerian graphs might ask whether a given diagram can be drawn without lifting one's pencil from the paper and without repeating any lines.
- **Königsberg Bridges Problem** (Euler): Can a pedestrian cross each of the seven bridges exactly once and return to his starting point?





- This is equivalent to asking whether the graph on the right has an Eulerian trail.

Existence of Cycles

Lemma

If G is a graph in which the degree of each vertex is at least 2, then G contains a cycle.

- If G has any loops or multiple edges, the result is trivial. We can therefore suppose that G is a simple graph. Let v be any vertex of G. We construct a walk $v \to v_1 \to v_2 \to \cdots$ inductively by choosing:
  - v_1 to be any vertex adjacent to v;
  - for each $i > 1$, v_{i+1} to be any vertex adjacent to v_i except v_{i-1}; the existence of such a vertex is guaranteed by our hypothesis.

  Since G has only finitely many vertices, we must eventually choose a vertex that has been chosen before. Suppose v_k is the first such vertex. Then the part of the walk lying between the two occurrences of v_k is the required cycle.

Euler's Characterization of Eulerian Graphs

Theorem (Euler 1736)

A connected graph G is Eulerian if and only if the degree of each vertex of G is even.

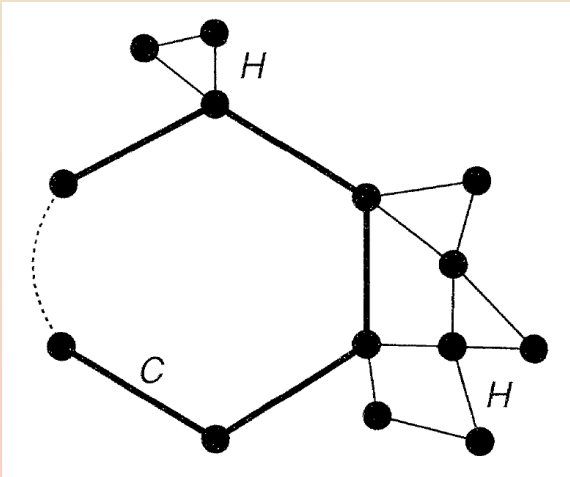
$\Rightarrow$: Suppose that P is an Eulerian trail of G. Whenever P passes through a vertex, there is a contribution of 2 towards the degree of that vertex. Since each edge occurs exactly once in P, each vertex must have even degree.

$\Leftarrow$: The proof is by induction on the number of edges of G. Suppose that the degree of each vertex is even. Since G is connected, each vertex has degree at least 2. So, by the lemma, G contains a cycle C.

- If C contains every edge of G, the proof is complete.
- If not, we remove from G the edges of C to form a new, possibly disconnected, graph H with fewer edges than G and in which each vertex still has even degree.

Euler's Characterization of Eulerian Graphs (Cont'd)

- We remove from G the edges of C to form a new, possibly disconnected, graph H with fewer edges than G and in which each vertex still has even degree. By the induction hypothesis, each component of H has an Eulerian trail. Since each component of H has at least one vertex in common with C, by connectedness, we obtain the required Eulerian trail of G as follows:



  - Trace the edges of C until a non-isolated vertex of H is reached;
  - Trace the Eulerian trail of the component of H that contains that vertex;
  - Continue along the edges of C until reaching a vertex belonging to another component of H, and so on.

The whole process terminates when we return to the initial vertex.

Consequence of Euler's Theorem

Corollary

A connected graph is Eulerian if and only if its set of edges can be split up into disjoint cycles.

Corollary

A connected graph is semi-Eulerian if and only if it has exactly two vertices of odd degree.

- Note that, in a semi-Eulerian graph, any semi-Eulerian trail must have one vertex of odd degree as its initial vertex and the other as its final vertex.
- Note also that, by the Handshaking Lemma, a graph cannot have exactly one vertex of odd degree.

Fleury's Algorithm

- We conclude with an algorithm for constructing an Eulerian trail in a given Eulerian graph.

Theorem (Fleury's Algorithm)

Let G be an Eulerian graph. Then the following construction is always possible, and produces an Eulerian trail of G.

Start at any vertex u and traverse the edges in an arbitrary manner, subject only to the following rules:

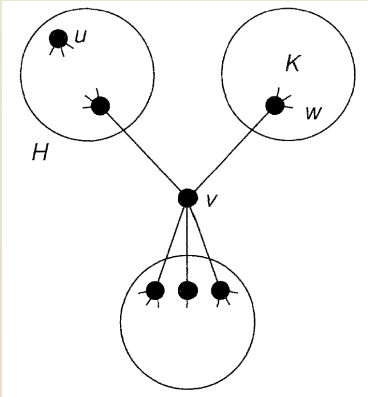
(i) Erase the edges as they are traversed, and if any isolated vertices result, erase them too;

(ii) At each stage, use a bridge only if there is no alternative.

- We show first that the construction can be carried out at each stage. Suppose that we have just reached a vertex v.
  - If $v \neq u$, then the subgraph H that remains is connected and contains only two vertices of odd degree, u and v.

Fleury's Algorithm (Cont'd)

- If $v \neq u$, then the subgraph H that remains is connected and contains only two vertices of odd degree, u and v.
  We must show that the removal of the next edge does not disconnect H, or, equivalently, that v is incident with at most one bridge.



If this is not the case, then there exists a bridge vw, such that the component K of $H - vw$ containing w does not contain u. Since the vertex w has odd degree in K, some other vertex of K must also have odd degree, yielding a contradiction.

- If $v = w$, the proof is almost identical, as long as there are still edges incident with u.

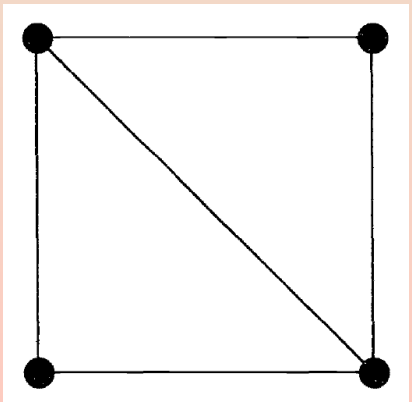
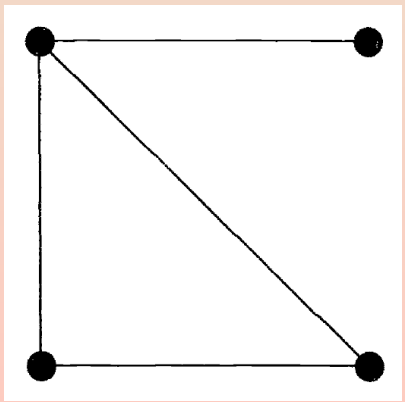
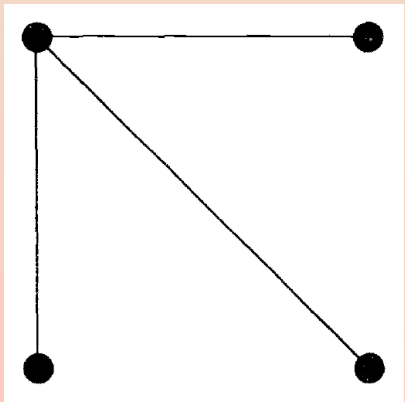
We show that this construction always yields a complete Eulerian trail: There can be no edges of G remaining untraversed when the last edge incident to u is used. Otherwise the removal of some earlier edge adjacent to one of these edges would have disconnected the graph, contradicting (ii).

Subsection 3

Hamiltonian Graphs

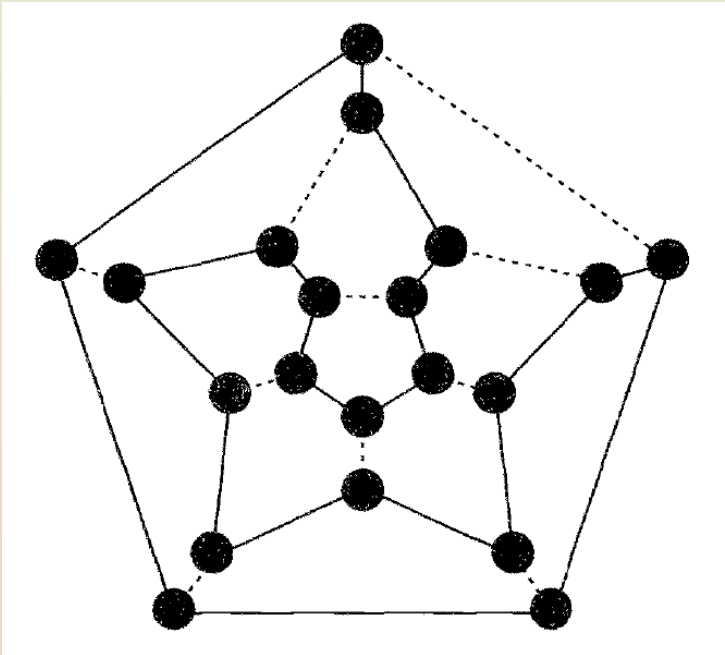
Hamiltonian and Semi-Hamiltonian Graphs

- In the previous section we discussed whether there exists a closed trail that includes every edge of a given connected graph G.
- A similar problem is to determine whether there exists a closed trail passing exactly once through each vertex of G.
- Such a trail must be a cycle, except when G is the graph N_1.
- Such a cycle is a **Hamiltonian cycle** and G is a **Hamiltonian graph**.
- A *non-Hamiltonian* graph G is **semi-Hamiltonian** if there exists a path passing through every vertex.
  Example: A Hamiltonian, semi-Hamiltonian and non-Hamiltonian graph, respectively.

Hamilton and the Dodecahedron Graph

- The name "Hamiltonian cycle" arises from the fact that Sir William Hamilton investigated the existence of such cycles in the dodecahedron graph:



- A Hamiltonian cycle in the dodecahedron graph is shown with heavy lines denoting its edges.

Ore's Theorem

- Finding necessary and sufficient conditions for a connected graph to be Hamiltonian is an open problem.

Theorem (Ore, 1960)

If G is a simple graph with n (≥ 3) vertices, and if $\deg(v) + \deg(w) \geq n$, for each pair of non-adjacent vertices v and w, then G is Hamiltonian.

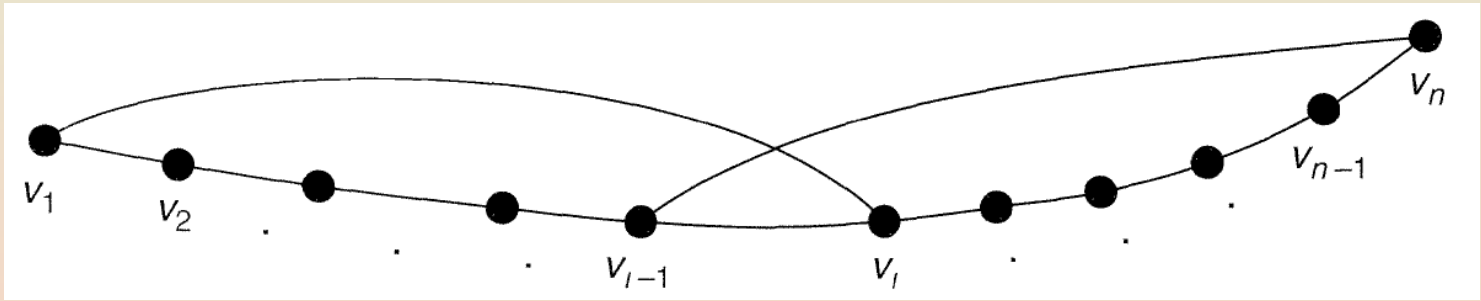
- We assume the theorem false, and derive a contradiction.

  Let G be a non-Hamiltonian graph with n vertices, satisfying the given condition on the vertex degrees. By adding extra edges if necessary, we may assume that G is "only just" non-Hamiltonian, in the sense that the addition of any further edge gives a Hamiltonian graph.

  Note: Adding extra edges does not violate the condition on the vertex degrees.

Ore's Theorem (Cont'd)

- It follows that G contains a path $v_1 \to v_2 \to \cdots \to v_n$ passing through every vertex. But since G is non-Hamiltonian, the vertices v_1 and v_n are not adjacent. So, by hypothesis, $\deg(v_1) + \deg(v_n) \geq n$. It follows that there must be some vertex v_i adjacent to v_1 with the property that v_{i-1} is adjacent to v_n.



But then the following is a Hamiltonian cycle:

$$v_1 \to v_2 \to \cdots \to v_{i-1} \to v_n \to v_{n-1} \to \cdots \to v_{i+1} \to v_i \to v_1$$

This contradicts the non-Hamiltonicity of G.

Dirac's Theorem

- Lacking a characterization of Hamiltonicity, the best we can do is formulate useful sufficient or useful necessary conditions.

Corollary (Dirac, 1952)

If G is a simple graph with n (≥ 3) vertices, and if $\deg(v) \geq \frac{n}{2}$, for each vertex v, then G is Hamiltonian.

- The result follows immediately from Ore's Theorem, since for each pair of vertices v and w (whether adjacent or not),

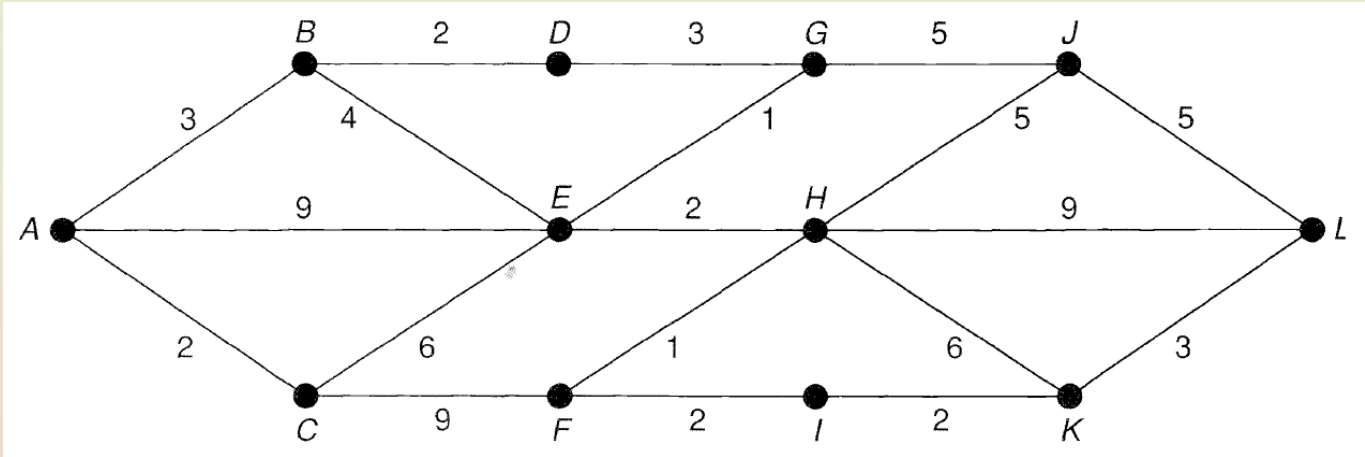
$$\deg(v) + \deg(w) \geq \frac{n}{2} + \frac{n}{2} = n.$$

Subsection 4

Some Algorithms

The Shortest Path Problem

- Suppose that we have a "map" in which the letters A-L refer to towns that are connected by roads:



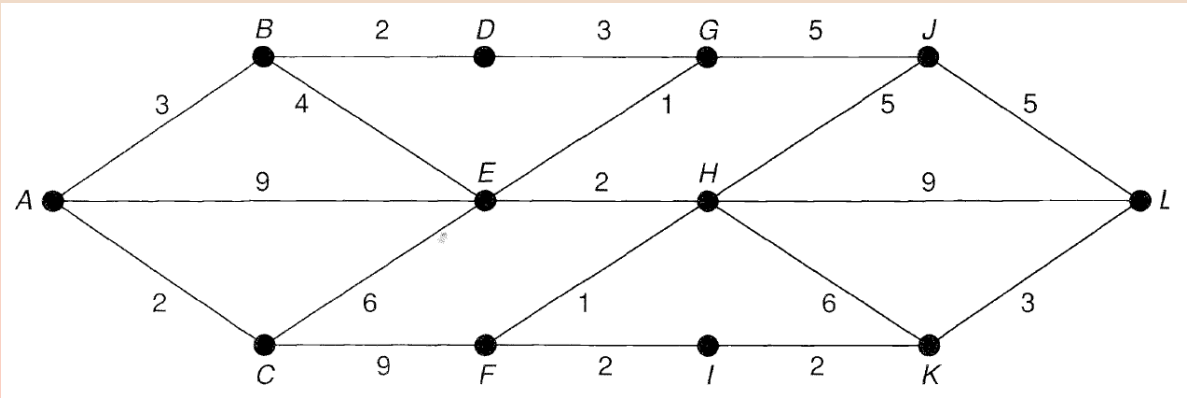
  If the lengths of these roads are as marked, we want to compute the length of the shortest path from A to L.

- Note that an upper bound for the answer can easily be obtained by taking any path from A to L and calculating its length.

  Example: The path $A \xrightarrow{3} B \xrightarrow{2} D \xrightarrow{3} G \xrightarrow{5} J \xrightarrow{5} L$ has total length 18. So the length of the shortest path cannot exceed 18.

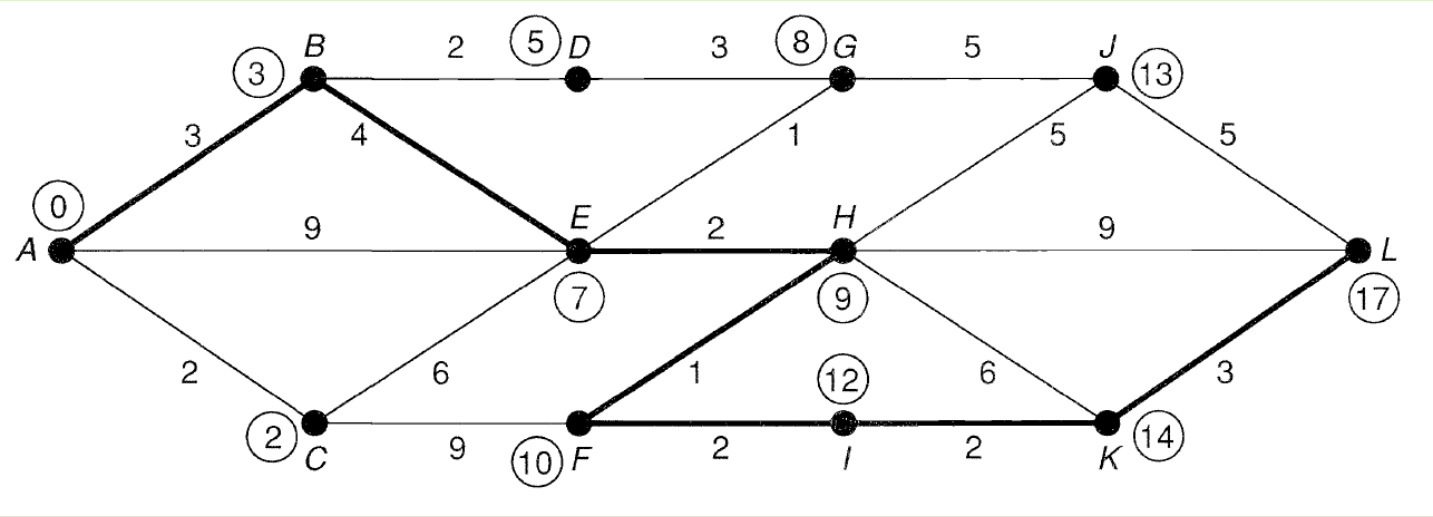
Weighted Graphs and Minimum-Weight Paths

- A connected graph in which a nonnegative number is assigned to each edge is called a **weighted graph**. The number assigned to each edge e is the **weight** of e, denoted by $w(e)$.
- The problem is to find a path from A to L with minimum total weight.
- If the weight of each edge is 1, the problem reduces to that of finding the number of edges in the shortest path from A to L.
- To solve this problem we move across the graph from left to right, associating with each vertex V a number $\ell(V)$ indicating the shortest distance from A to V.



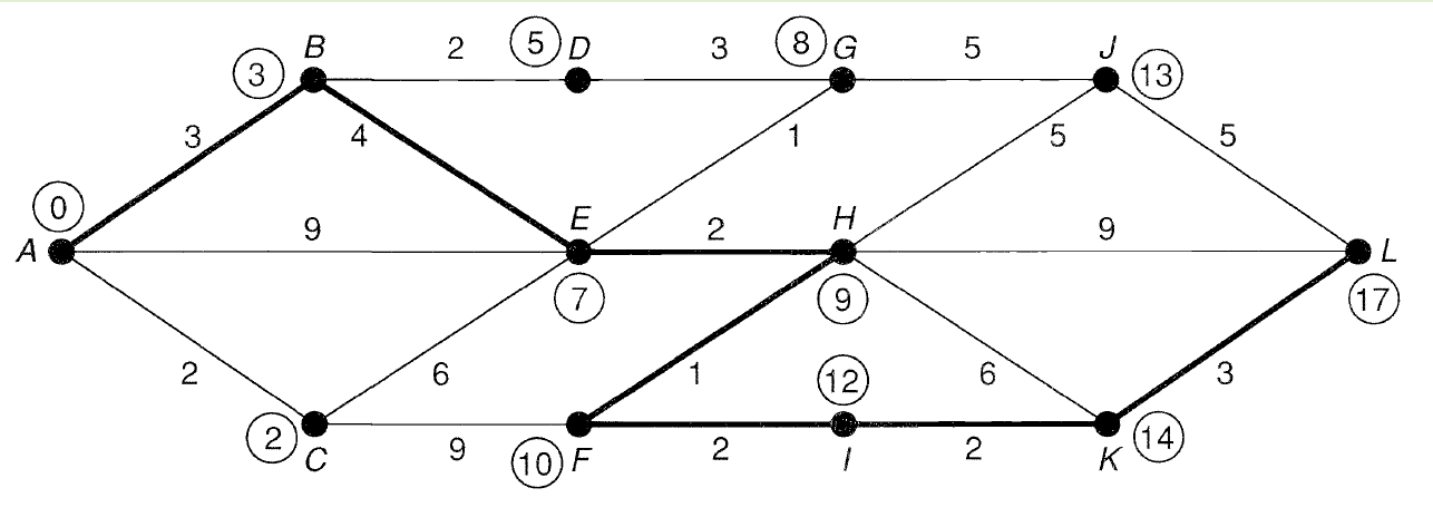
- E.g., when we reach K, $\ell(K)$ is the smallest of $\ell(H)+6$ and $\ell(I)+2$.

Illustrating the Algorithm



- We first assign A the label 0.
- Adjacent to A:
  - $\ell(B) = \ell(A) + 3 = 3$;
  - $\ell(E) = \ell(A) + 9 = 9$;
  - $\ell(C) = \ell(A) + 2 = 2$.
    $\ell(C) = 2$ is the smallest label.
- Adjacent to C:
  - $\ell(F) = \ell(C) + 9 = 1$;
  - $\ell(E) = \ell(C) + 6 = 8 < 9$.
    $\ell(B) = 3$ is the smallest label.

Illustrating the Algorithm (Cont'd)



- Adjacent to B:
  - $\ell(D) = \ell(B) + 2 = 5$;
  - $\ell(E) = \ell(B) + 4 = 7 < 8$.

    $\ell(D) = 5$ is the smallest label.
- Continuing in this way, we successively obtain the permanent labels

$$\ell(E) = 7,\ \ell(G) = 8,\ \ell(H) = 9,\ \ell(F) = 10,\ \ell(I) = 12,\ \ell(J) = 13,\ \ell(K) = 14,\ \ell(L) = 17.$$

- It follows that the shortest path from A to L has length 17.

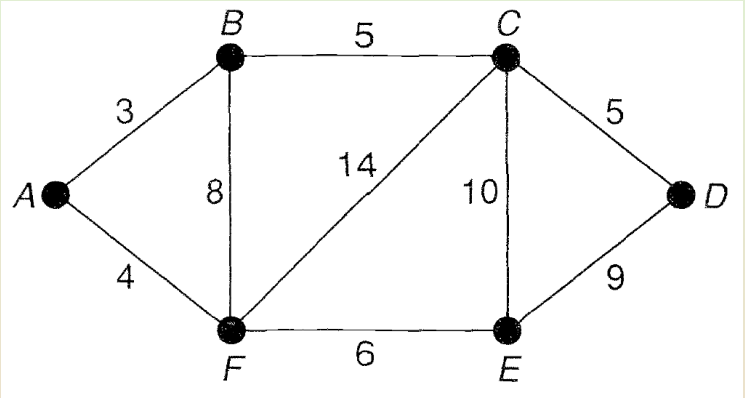
The Chinese Postman Problem

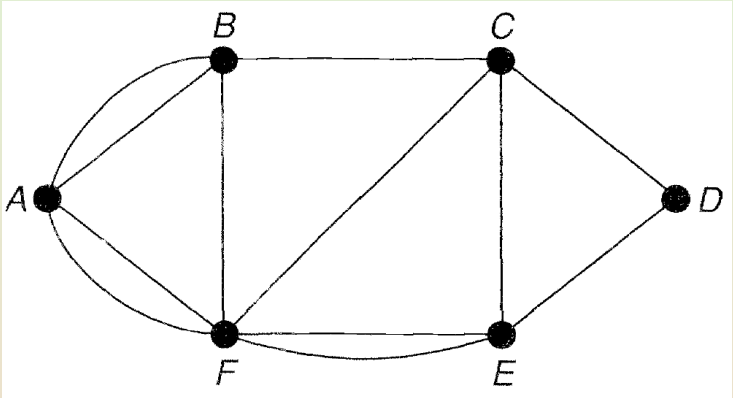
- A postman wishes to deliver his letters, covering the least possible total distance and returning to his starting point.

  He must obviously traverse each road in his route at least once, but should avoid covering too many roads more than once.
- We consider a weighted graph corresponding to the network of roads, with the weight of each edge being the length of the corresponding road.
- The requirement is to find a closed walk of minimum total weight that includes each edge at least once.
  - If the graph is Eulerian, then any Eulerian trail is a closed walk of the required type.
    Such an Eulerian trail can be found by Fleury's algorithm.
  - If the graph is not Eulerian, then the problem is harder, but there are still efficient algorithms for its solution.

The Chinese Postman Problem: A Special Case

- Suppose exactly two vertices have odd degree:



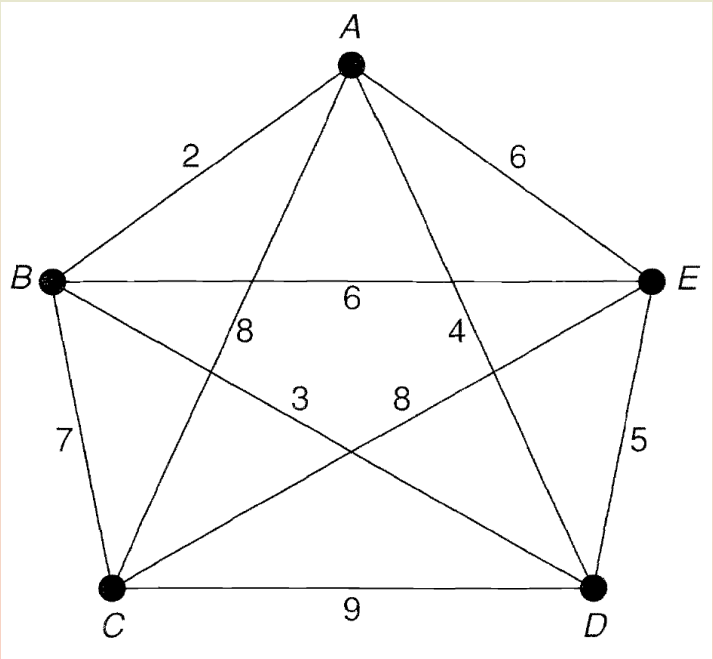


  - Since vertices B and E are the only vertices of odd degree, we can find a semi-Eulerian trail from B to E covering each edge exactly once.
  - In order to return to the starting point, covering the least possible distance, we now find the shortest path from E to B using the algorithm described previously.
  - The solution of the Chinese postman problem is then obtained by taking this shortest path together with the original semi-Eulerian trail.

Note: If we combine the shortest path and the semi-Eulerian trail, we get an Eulerian graph as on the right figure.

The Traveling Salesman Problem

- A travelling salesman wishes to visit several given cities and return to his starting point, covering the least possible total distance.
  Example: If there are five cities A, B, C, D and E, and if the distances are as in the figure



then the shortest possible route is $A \xrightarrow{2} B \xrightarrow{3} D \xrightarrow{5} E \xrightarrow{8} C \xrightarrow{8} A$.
Thus, we get total distance of 26.

The Traveling Salesman Problem (Cont'd)

- This problem can also be reformulated in terms of weighted graphs.

  In this case, the requirement is to find a Hamiltonian cycle of least possible total weight in a weighted complete graph.
- One possible algorithm is to calculate the total distance for all possible Hamiltonian cycles, but this is far too complicated for even a moderate number of cities:

  For 20 cities, then the number of possible cycles is $\frac{19!}{2} \approx 6 \times 10^{16}$.
- Various other algorithms have been proposed, but they take too long to apply.
- There are several heuristic algorithms that quickly tell us approximately what the shortest distance is.