

Complex Networks:

Lecture 3: Introduction to Graph Theory

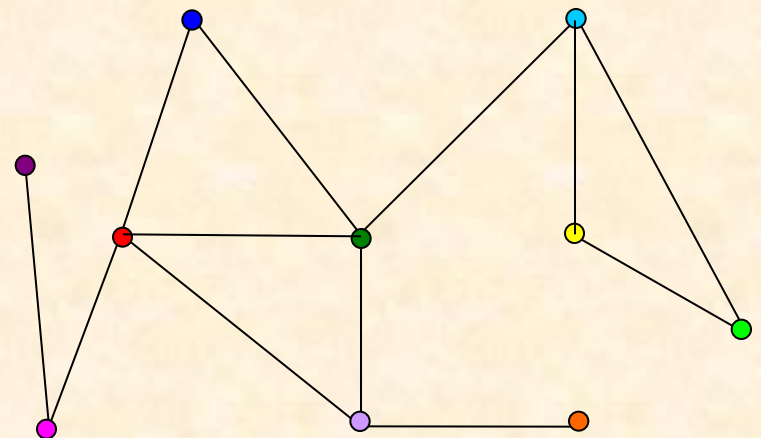
EE 6605

Instructor: G Ron Chen



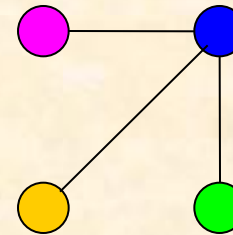
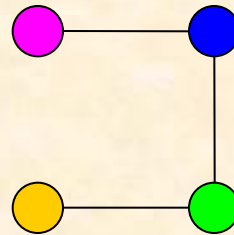
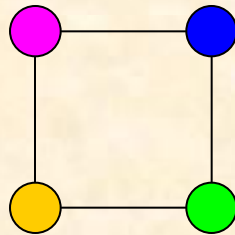
Review

- **Walk:** A finite sequence of edges, one after another, in the form of $v_1v_2, v_2v_3, \dots, v_{n-1}v_n$ where $N(G) = \{v_1, v_2, \dots, v_n\}$ are nodes.
- A walk is denoted by $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n$ and the number of edges in a walk is called its **length**.
- **Trail:** A walk in which all edges are distinct.
- **Path:** A trail in which all nodes are distinct, except perhaps $v_1 = v_n$ which is called a **closed path**, often called a **circuit** (or, sometimes, a **loop** or a **cycle**).
- **Tree:** A graph with no circuits.



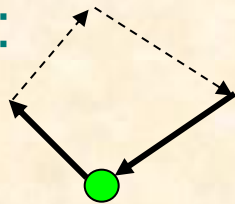
Eulerian Graphs

- **Eulerian graph:** If it has a closed trail that traverses each edge once and once only.
- There may be more than one such trail, each of which is called an **Eulerian trail**.
- **Semi-Eulerian graph:** If it has a trail, need not be closed, that traverses each edge once and once only.



Main Results

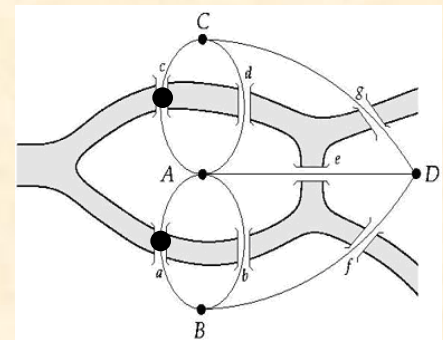
- **Theorem:** *A connected graph is Eulerian if and only if the degree of every node in the graph is an even number.*
- **Proof:** If a node is a starting point, then the trail has to return to it:



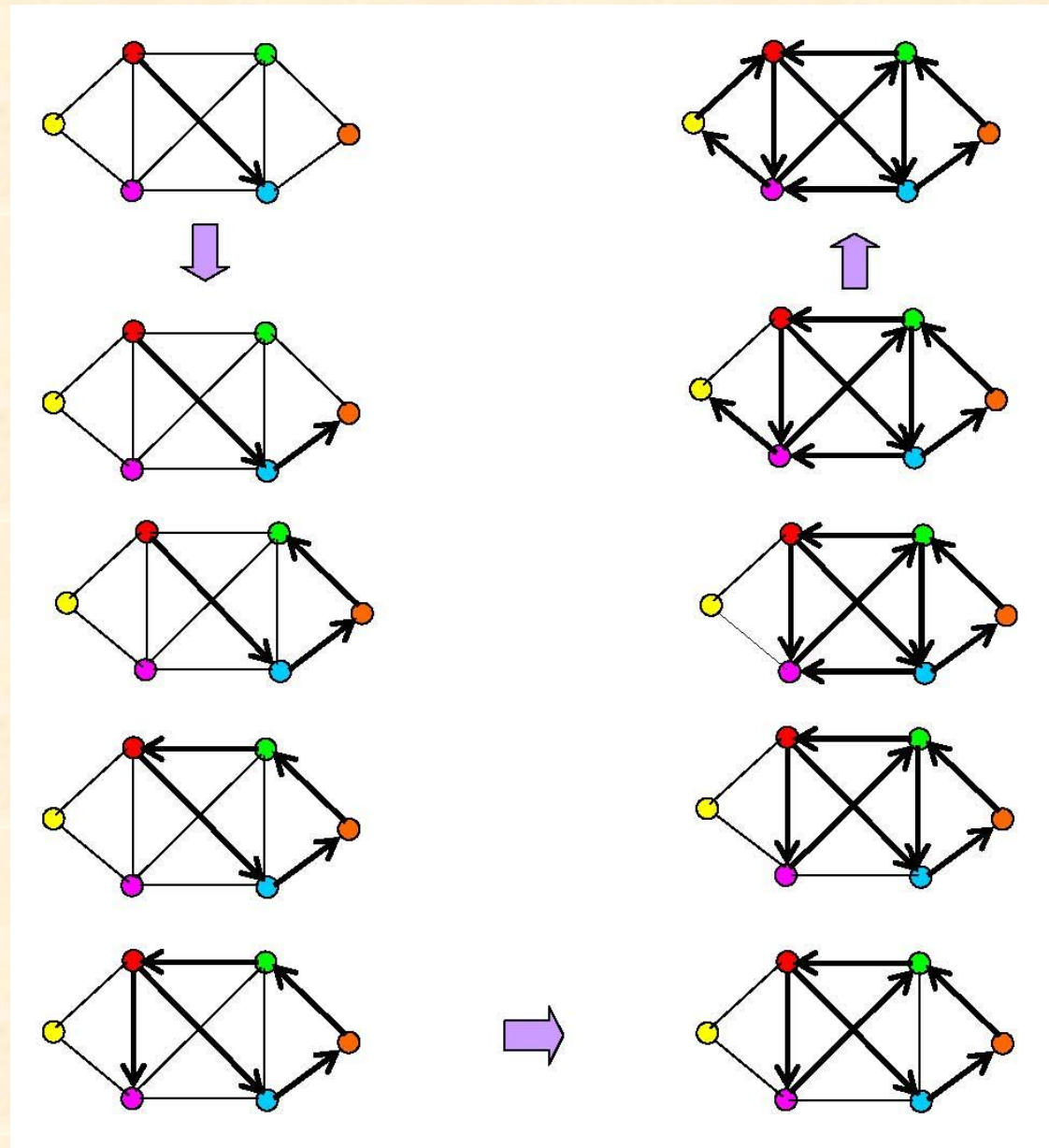
If a node has a trail enter it, then the trail has to leave it:



- **Corollary:** *The seven-bridge problem of Königsburg has **no** solutions.*



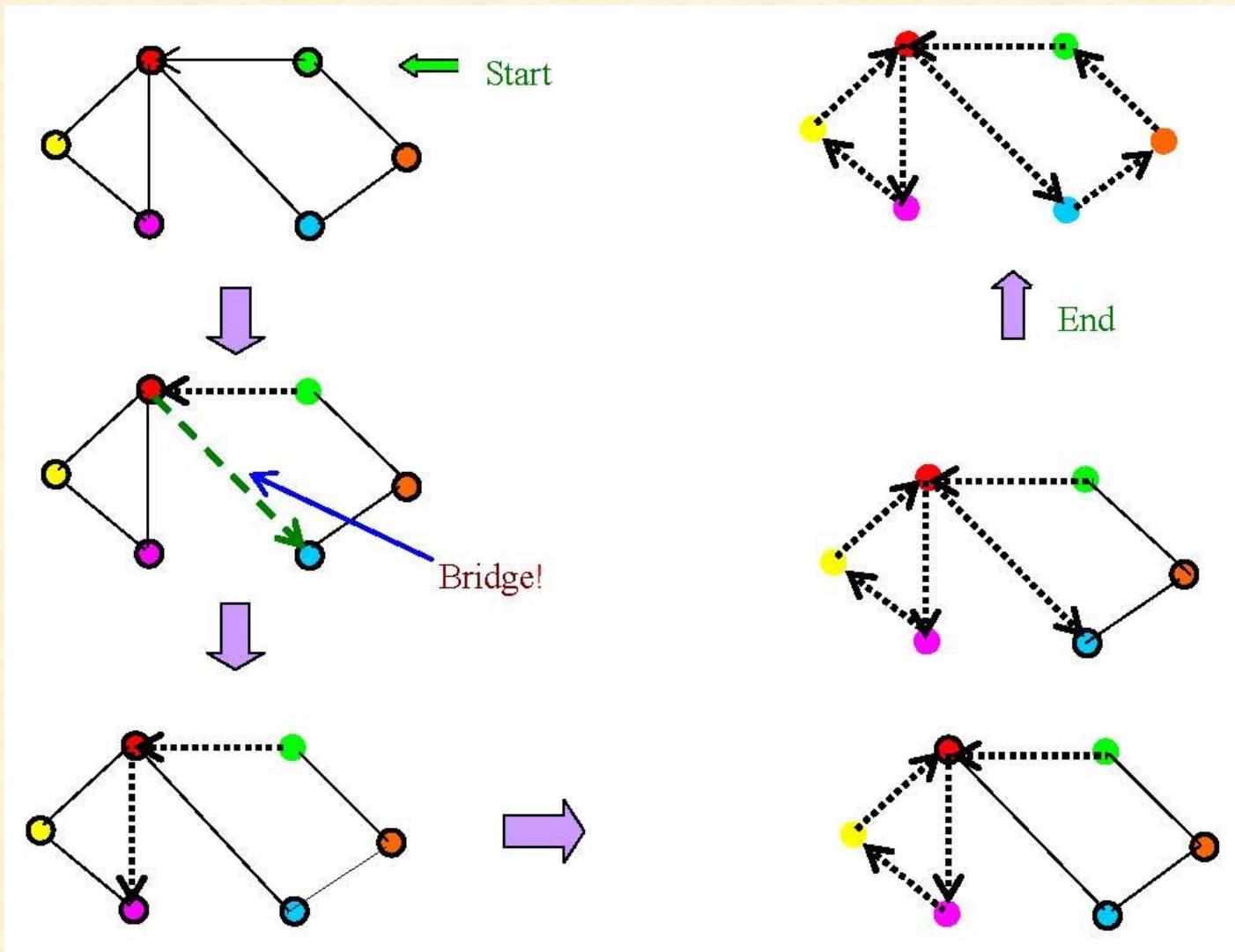
Example:



Algorithm to find Eulerian Graphs

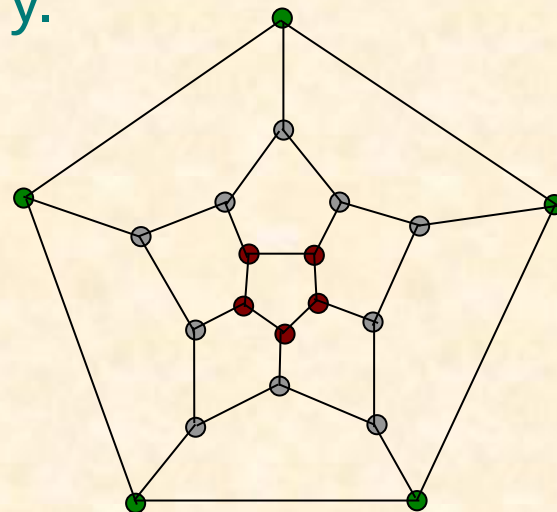
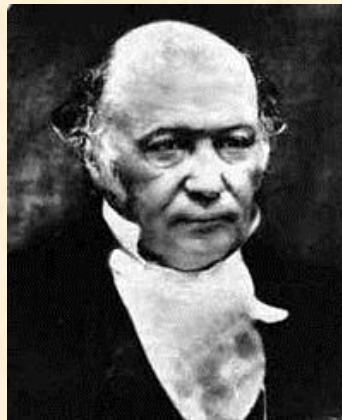
- **Corollary** (Fleury Algorithm) *In any given Eulerian graph G , an Eulerian trail can be found by the following procedure:*
- *Start from any node in G . Walk along the edges of G in an arbitrary manner, subject to the following rules:*
 - *erase the edges as they are traversed;*
 - *erase the resulting isolated nodes;*
 - *walk through a bridge only if there are no other alternatives.*

Example:

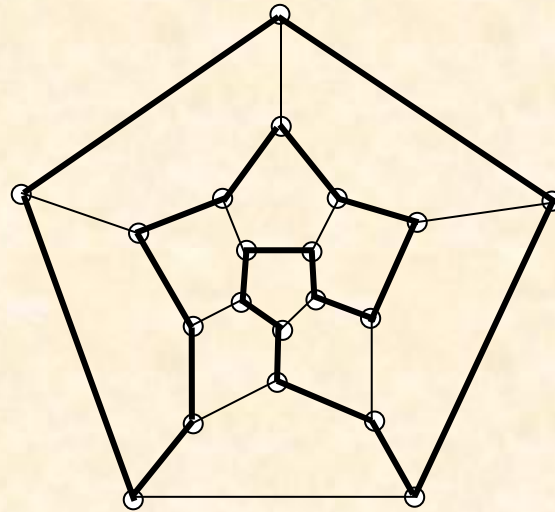


Sir William Rowan Hamilton (1805-1865)

In 1856, the English mathematician William R. Hamilton studied the world navigation problem and considered a map with 20 nodes representing cities connected by sailing routes, as depicted by the following figure. He wanted to find out if one can traverse through every city once and once only, and finally return to the starting city. His study eventually led to the establishment of the now-famous Hamiltonian graph theory.

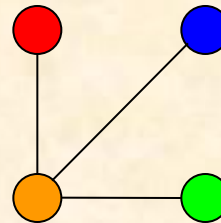
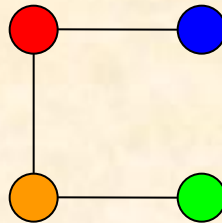
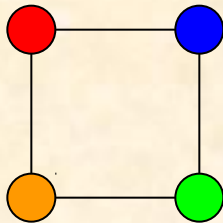


One solution



Hamiltonian Graphs

- **Hamiltonian graph:** If it has a **closed trail** that traverses each node **once and once only**.
- A trivial case is a single node and a nontrivial Hamiltonian graph must be a circuit, called a **Hamiltonian circuit**.
- **Semi-Hamiltonian graph:** If it has a trail, **need not be closed**, that traverses each node **once and once only**.



Some Results

Theorem: Let G be a simple graph with $N (\geq 3)$ nodes. If, for every pair of non-adjacent nodes v and u , their degrees always satisfy $k(v) + k(u) \geq N$, then G is Hamiltonian.

Proof. Consider any path in the graph, for example:

$$v_1 - v_2 - v_3$$

Since $k(v_1) + k(v_3) \geq 3$, either v_1 or v_3 has another neighbor v_i , for example,

$$\begin{array}{c} v_i \\ | \end{array}$$

$$v_1 - v_2 - v_3$$

But then $k(v_1) + k(v_i) \geq 3$, so either v_1 or v_i has another neighbor, ...

Thus, the only way to satisfy $k(v_1) + k(v_i) \geq 3$ for non-adjacent v_1 and v_i is to close the loop:

$$\begin{array}{ccc} & - \dots & v_i \\ | & & | \\ v_1 - v_2 - v_3 & & \end{array}$$

This gives a Hamiltonian circuit.

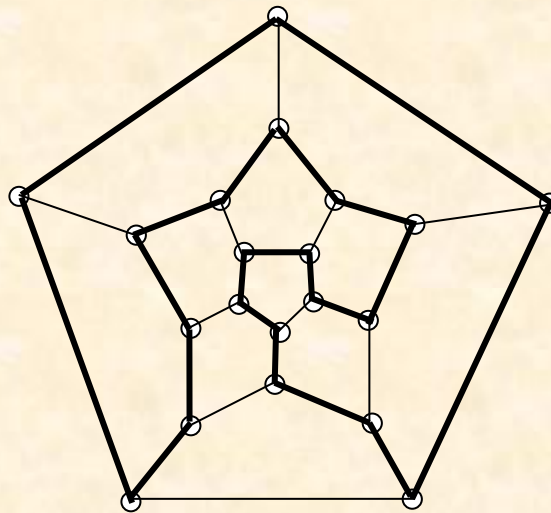
Corollary: Let G be a simple graph with $N(\geq 3)$ nodes. If every pair of nodes has degree $k(v) + k(u) \geq N$, then G is Hamiltonian.

Corollary Let G be a simple graph with $N(\geq 3)$ nodes. If every node has degree $k \geq N/2$, then G is Hamiltonian.

The above conditions are sufficient but not necessary

For example:

$$N = 20$$



$$k(v) + k(u) < N$$

$$k(v) < N/2$$

Knight's Tour Game

Can the knight visit every block, once and once only, and finally return to the starting point?



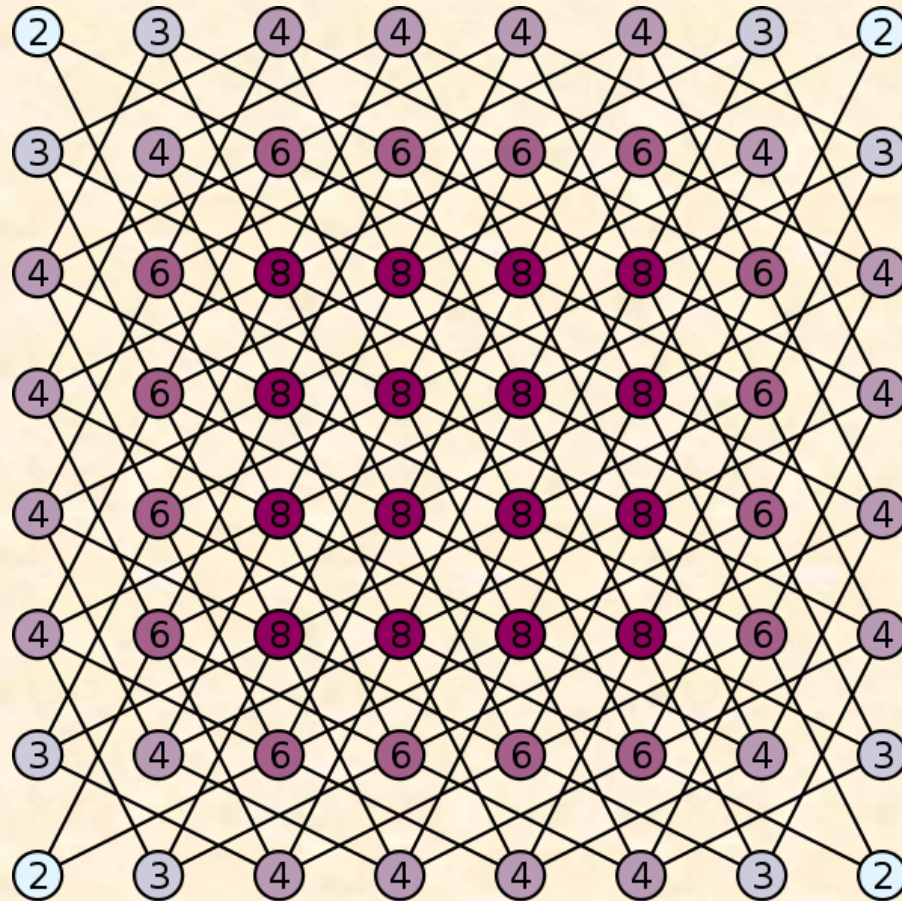
= Is this a Hamiltonian graph ?

Knight's Tour Game

Can the knight visit every block, once and once only, and finally return to the starting point?



5 X 5 has open solution (semi-Hamiltonian)
but no closed solution (Hamiltonian)



8 X 8 – Yes, it always has a closed solution

Namely, it is Hamiltonian, proved by Euler in 1759

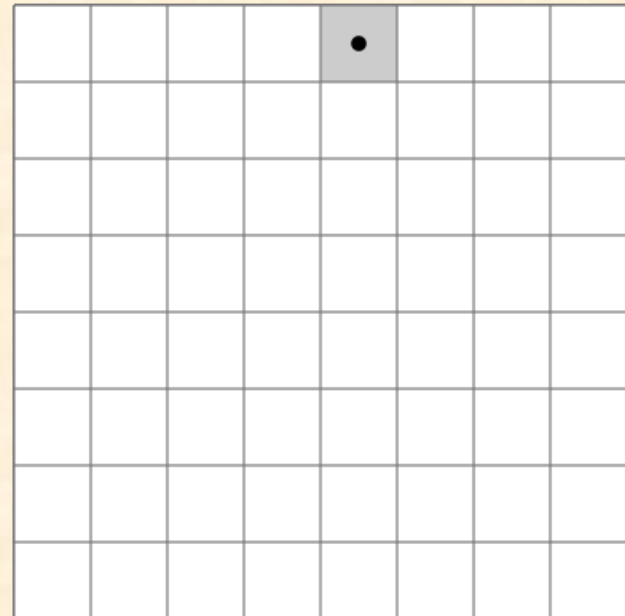
Theorem (Cull and de Curtins, 1978) On any rectangular board whose smaller dimension is at least 5, there is a (possibly open) knight's tour (semi-Hamiltonian)

Theorem (Schwenk, 1991) For any $m \times n$ board with $m \leq n$ a closed knight's tour is always possible (= Hamiltonian), except the following few cases:

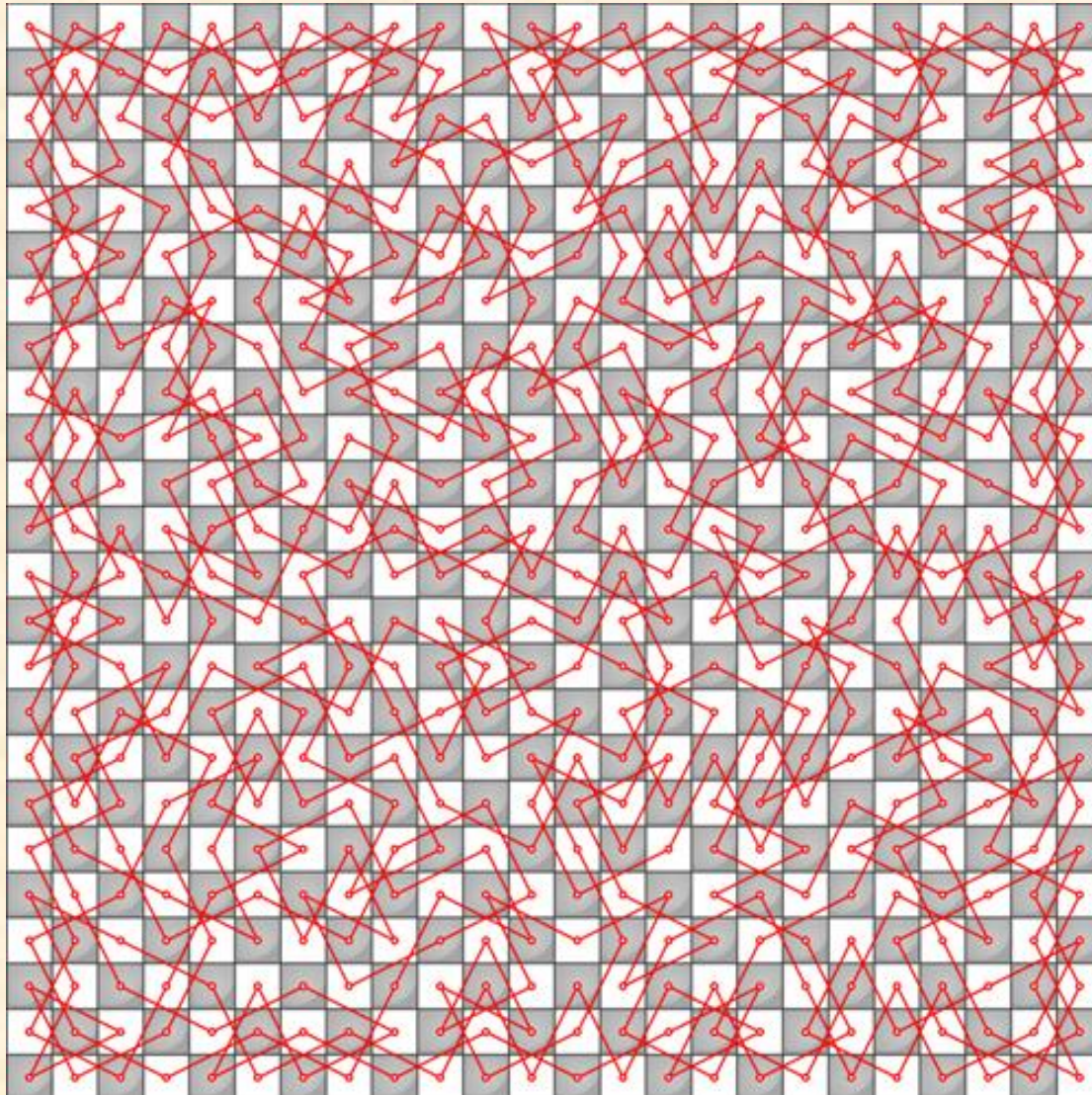
- (1) m and n are both odd
- (2) $m = 1, 2$, or 4
- (3) $m = 3$ and $n = 4, 6$, or 8

[Wiki](#)

8x8

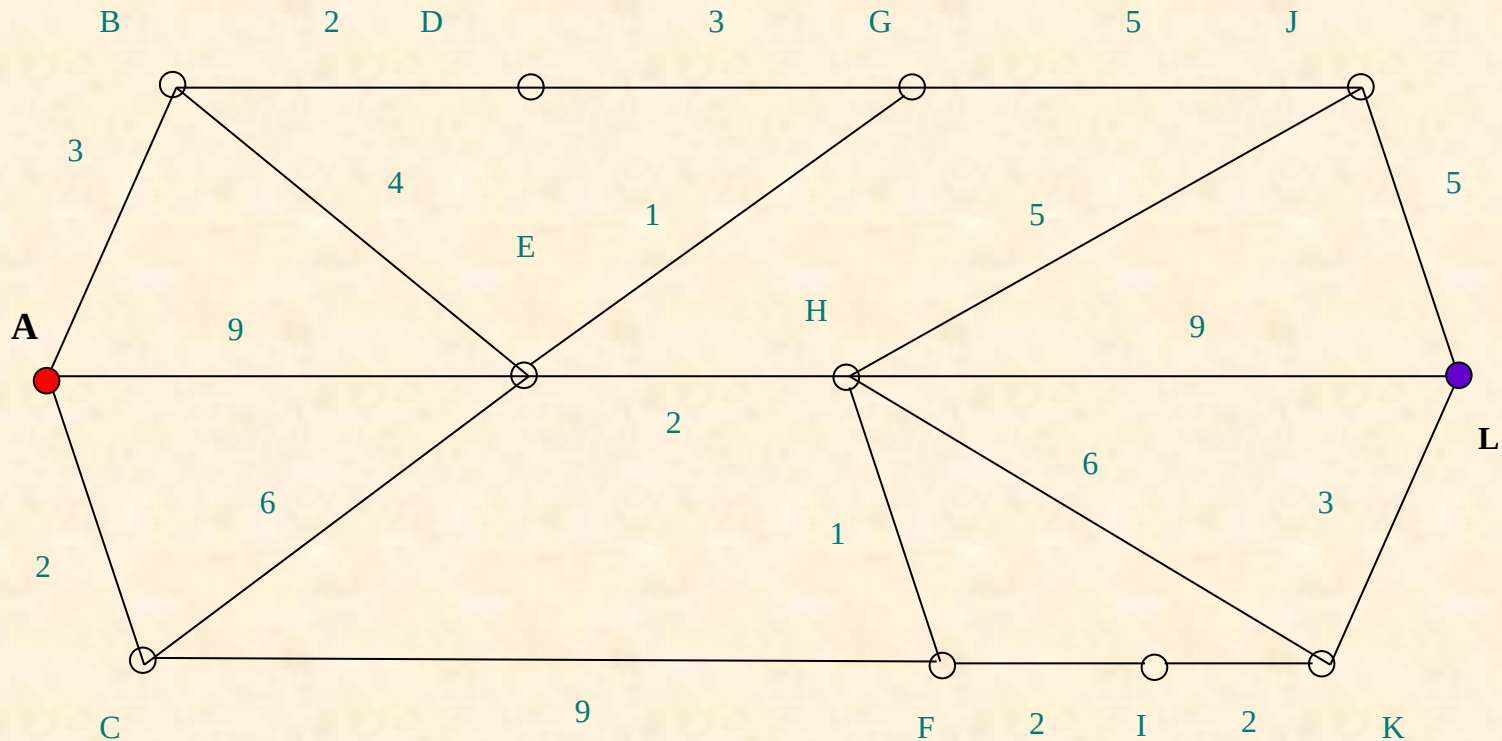


24 x 24



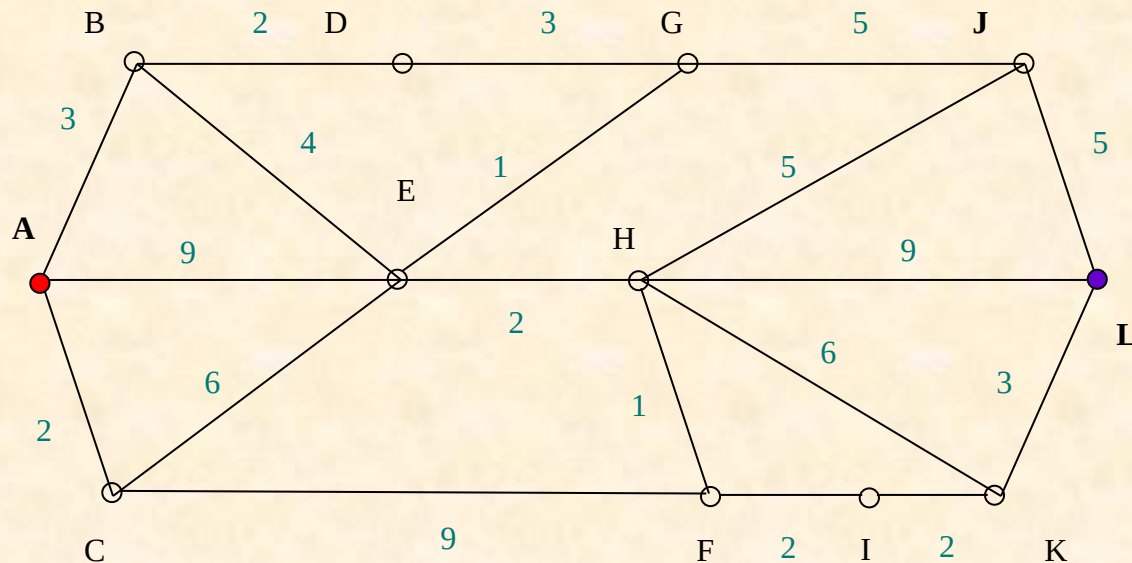
Shortest Path Length Problem

Q: What is the shortest path length from *A* to *L* ?



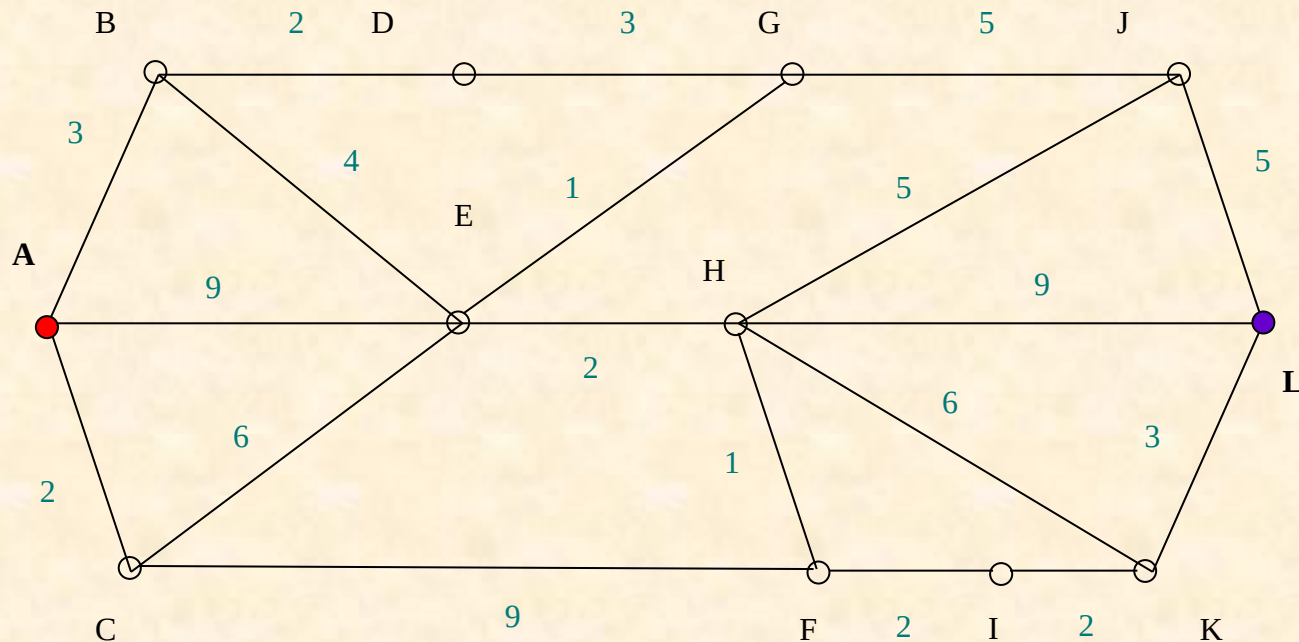
Approach to Solving the Shortest Path Length Problem

- **Approach:** Moving from A toward L and associating each intermediate node V with a number $l(V)$ that is equal to the shortest path length from A to V .
- **For example:** from A to J , one has $l(J) = l(G) + 5$ or $l(J) = l(H) + 5$ whichever is shorter.



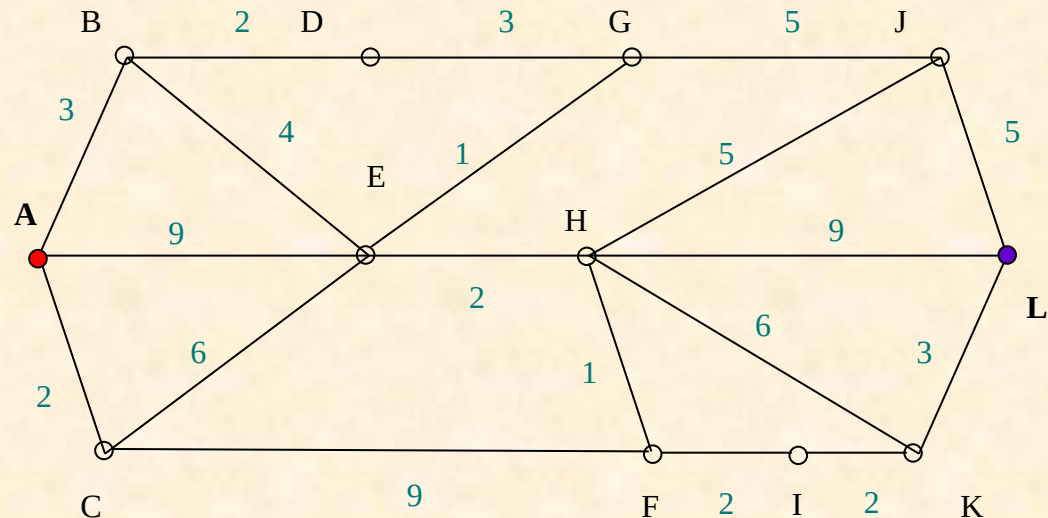
Approach to Solving the Shortest Path Length Problem

- Starting from A with $l(A) = 0$, moving from A toward L step by step, one has
$$l(B) = l(A) + 3 = 3 \quad l(E) = l(A) + 9 = 9 \quad l(C) = l(A) + 2 = 2$$
- Therefore, node B is chosen, with $l(B) = 3$, node C is chosen, with $l(C) = 2$ and node E is chosen with $l(E) = 9$



Approach to Solving the Shortest Path Length Problem

- Looking at the nodes adjacent to B , one has
$$l(D) = l(B) + 2 = 5 \quad l(E) = l(B) + 4 = 7$$
- Looking at the nodes adjacent to C , one has
$$l(E) = l(C) + 6 = 8 \quad l(F) = l(C) + 9 = 11$$
- So, $l(D) = 5$ is uniquely determined.
- But $l(E) = 7$, or 8 or 9. Hence, node E is chosen with $l(E) = l(B) + 4 = 7$
$$l(H) = l(E) + 2 = 9$$
- F is chosen with $l(F) = 11$
- But also $l(F) = l(H) + 1$ so F is chosen with $l(F) = l(H) + 1 = 10$



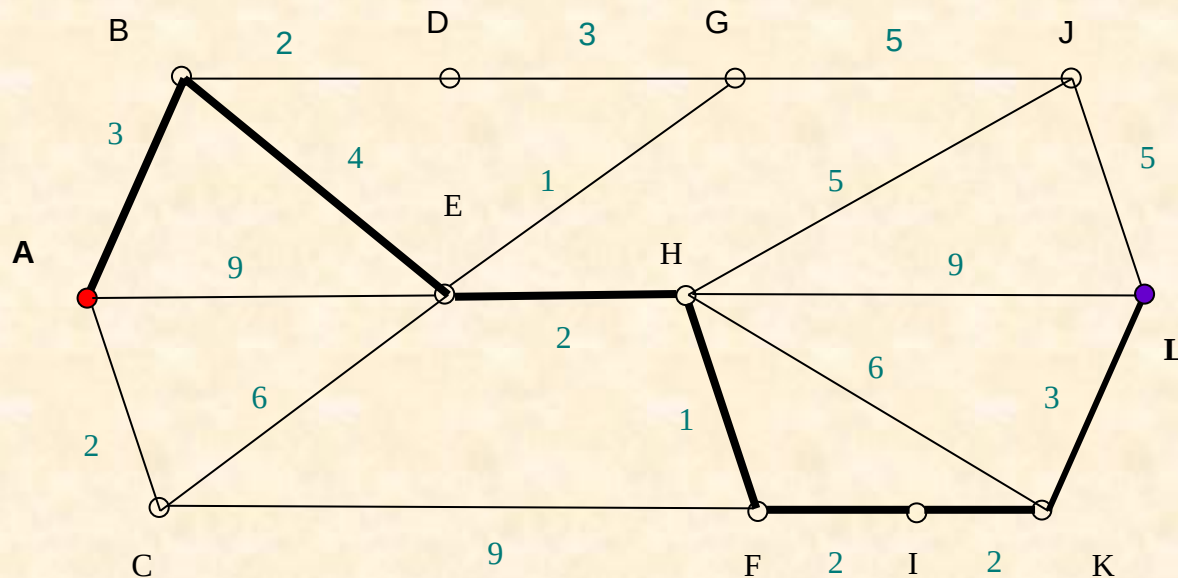
Approach to Solving the Shortest Path Length Problem

- Continuing this way, one obtains

$$l(A)=0 \quad l(B)=3 \quad l(C)=2 \quad l(D)=5 \quad l(E)=7 \quad l(F)=10$$

$$l(G)=8 \quad l(H)=9 \quad l(I)=12 \quad l(J)=13 \quad l(K)=14 \quad l(L)=17$$

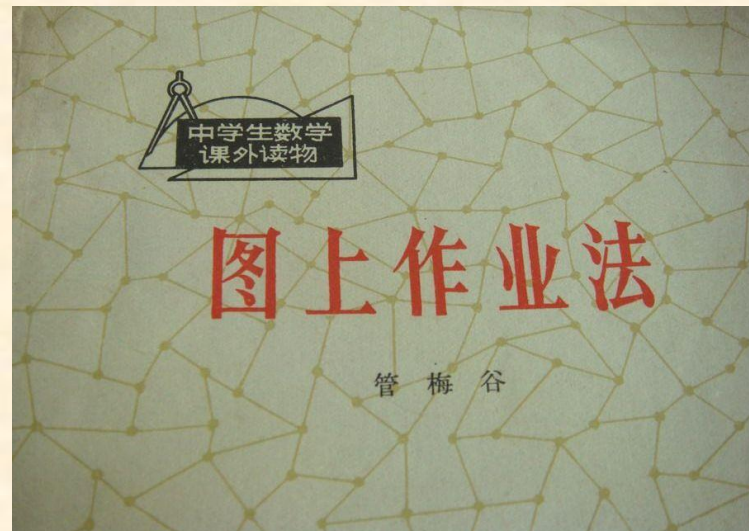
- Final result: total shortest path length = 17 from A to L :



- Solutions may not be unique, but all will be shortest

Chinese Postman Problem

- ❖ The problem is for a postman to deliver all letters in such a way that he passes every street at least once and finally returns to the starting point (the post office), traversing a shortest possible total path-length.
- ❖ It differs from an Eulerian graph in that it allows multiple edges between two nodes. Also, it has a requirement of having minimal path-length.
- ❖ First solution given by
Mei-ko Kwan (1934—)
(管梅谷 from Shanghai)



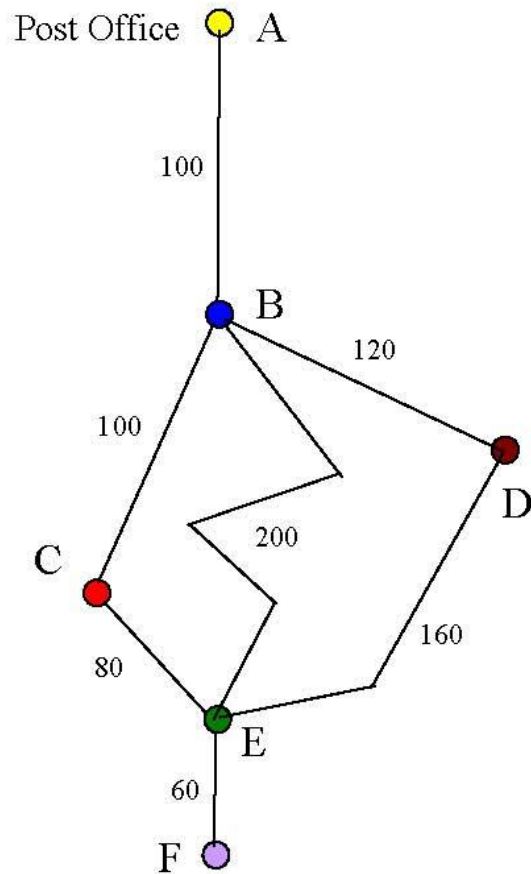
Chinese Postman Problem

❖ Kwan's Algorithm

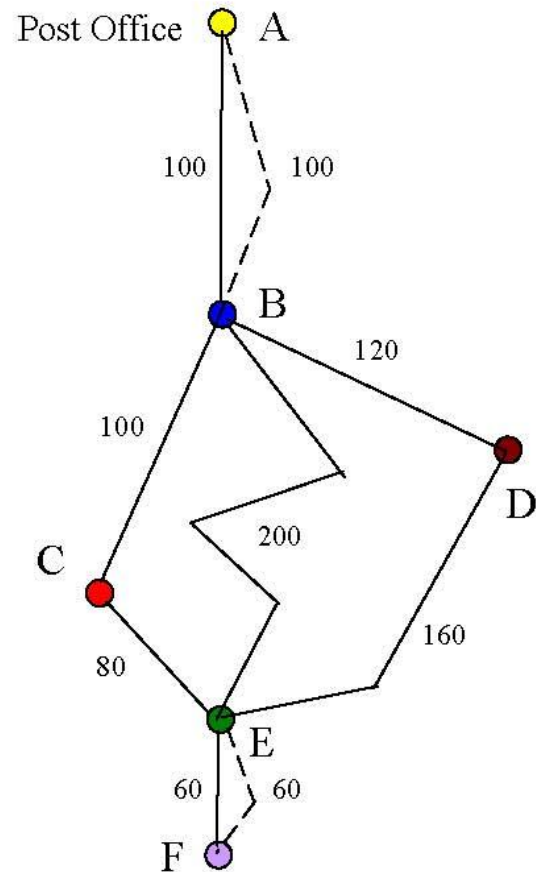
In any given connected graph G , a Chinese postman walk can be found by the following off-line procedure

- (1) Identify the initial node in G , then do the following:
 - ❖ for each odd-degree node in G , add an edge to connect it to one of its neighboring node
 - ❖ repeat the above, until all nodes in G have **even** degrees.
- (2) Check **all** resulting reconfigurations -- **if** the increment of the total path-length of **every** loop so **created** is not longer than one half of the total path-length of the corresponding **original** loop, **then** keep this loop as a solution.

Example:

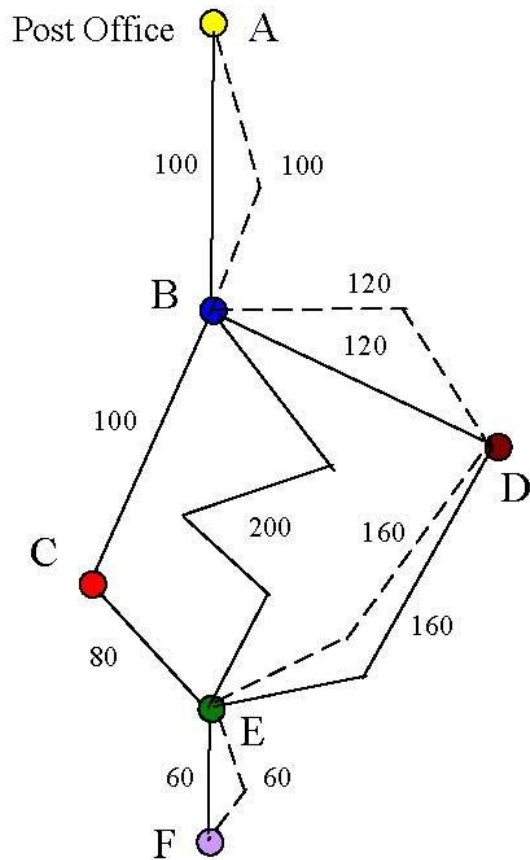


(a) The routing map

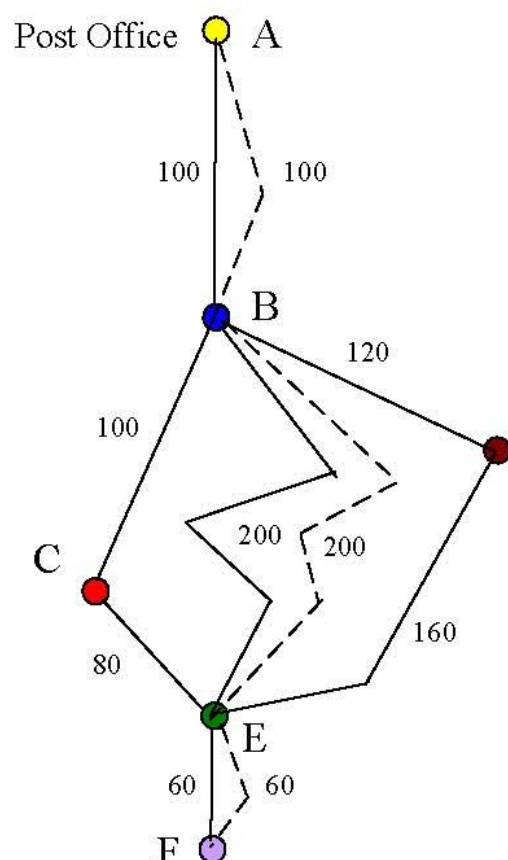


(b) Making nodes A and F be of even degree

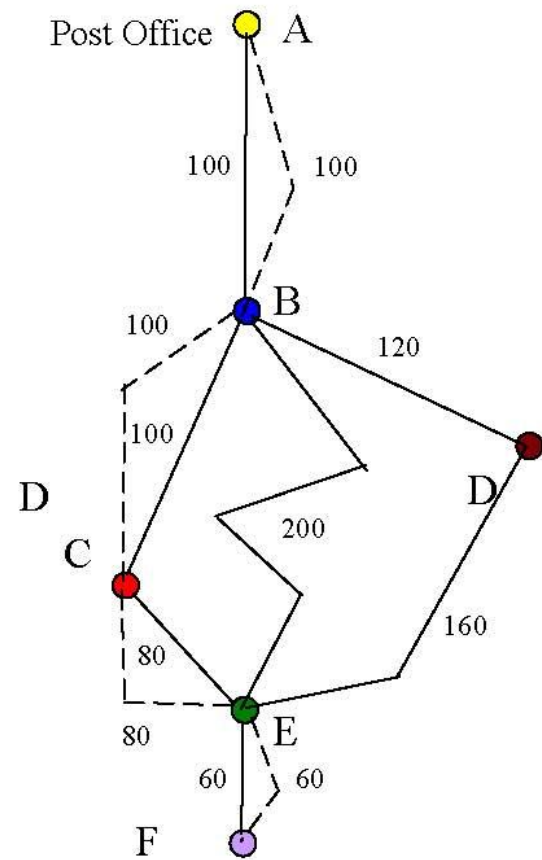
Example:



(c) Possible route 1



(d) Possible route 2



(e) Possible route 3

Through new dashed edges

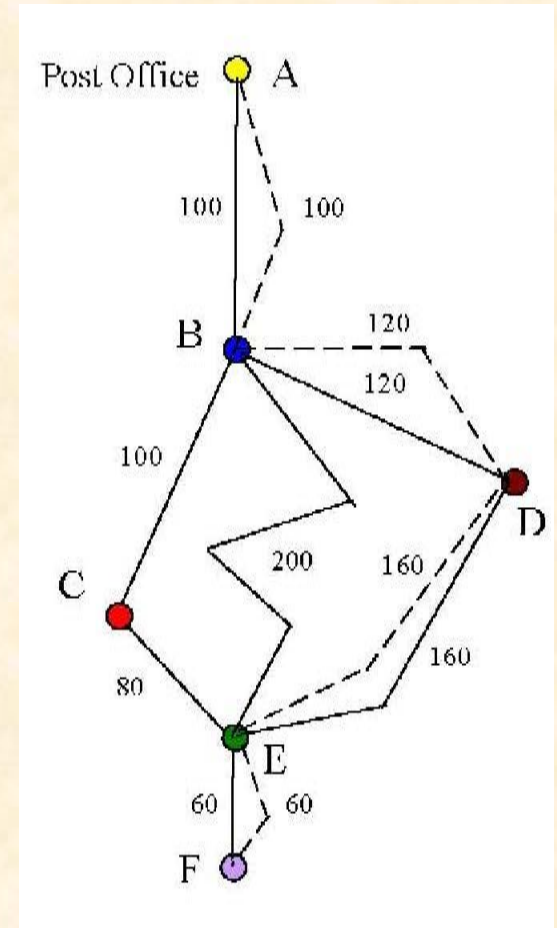
Example:

There are 3 existing loops:

- (1) Total path-length of B-D-E-C-B = $120 + 160 + 80 + 100 = 460$
- (2) Total path-length of B-C-E-B = $100 + 80 + 200 = 380$ [this is unrelated with increment]
- (3) Total path-length of B-D-E-B = $120 + 160 + 200 = 480$

Possible choices of routes are:

Fig. (c): New increment of total path-length = $120 + 160 = 280$, which is longer than one half of the corresponding loop B-D-E-C-B = $460/2 = 230$, and also longer than one half of the corresponding loop B-D-E-B = $480/2 = 240$.
→ Not a solution.



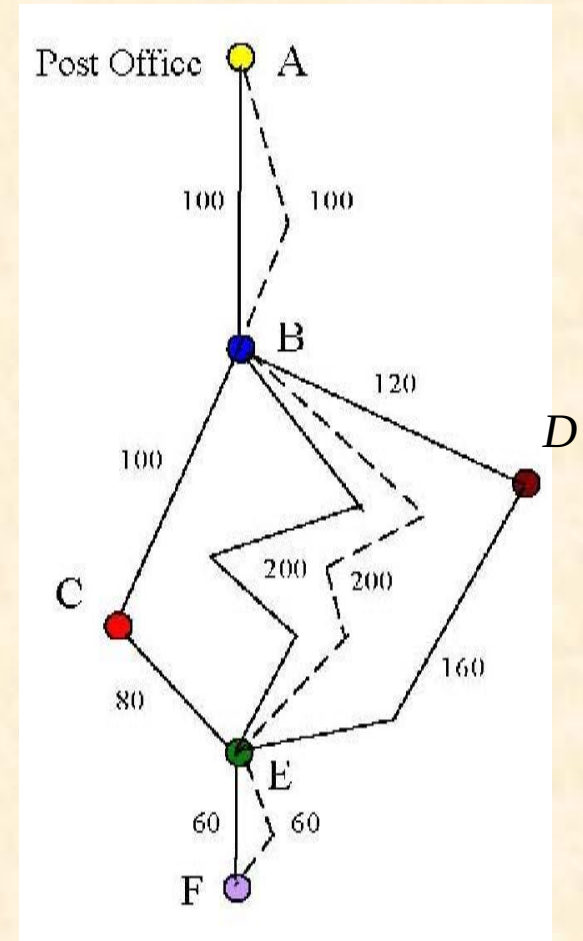
Example:

There are 3 existing loops:

- (1) Total path-length of B-D-E-C-B = $120 + 160 + 80 + 100 = 460$ [this is unrelated with increment]
- (2) Total path-length of B-C-E-B = $100 + 80 + 200 = 380$
- (3) Total path-length of B-D-E-B = $120 + 160 + 200 = 480$

Possible choices of routs are:

Fig. (d): New increment of total path-length = 200, which is longer than one half of the corresponding loop B-C-E-B = $380/2 = 190$ (although not longer than one half of the corresponding loop B-D-E-B = $480/2 = 240$). → Not a solution.



Example:

There are 3 existing loops:

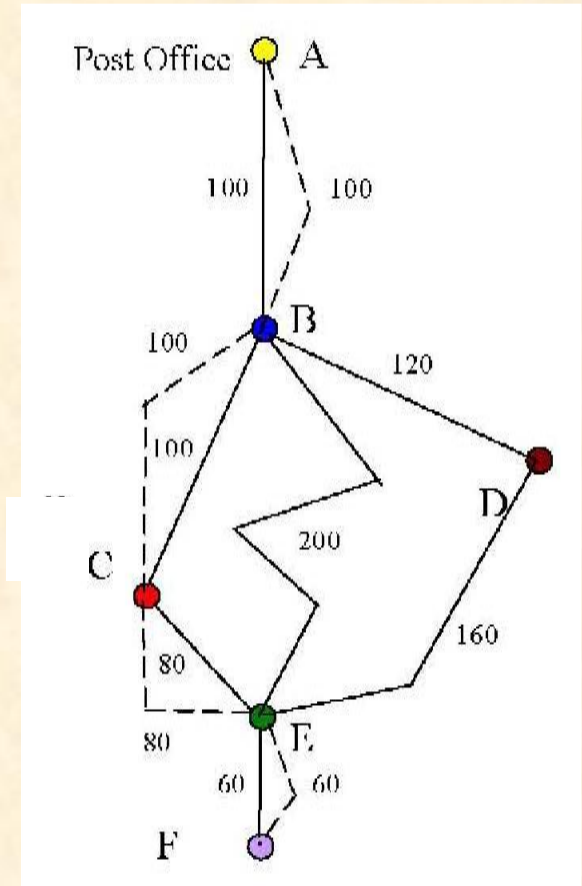
- (1) Total path-length of B-D-E-C-B = $120 + 160 + 80 + 100 = 460$
- (2) Total path-length of B-C-E-B = $100 + 80 + 200 = 380$
- (3) Total path-length of B-D-E-B = $120 + 160 + 200 = 480$ [this is unrelated with increment]

Possible choices of routs are:

Fig. (e): New increment of total path-length = $100 + 80 = 180$, which is shorter than one half of the corresponding loop B-C-E-B = $380/2 = 190$, and also shorter than one half of the corresponding loop B-C-E-D-B = $460/2 = 230$.

This is a solution → A-B-C-E-F-E-D-B-E-C-B-A

(Note: solution may not be unique in general)

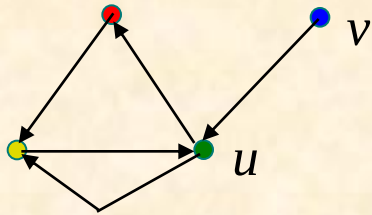


BREAK

10 minutes

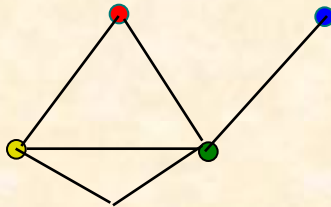
Directed Graphs

Digraph (= directed graph): edge (u,v) : $u \leftarrow v$



simple digraph

Underlying graph:

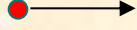


non-simple graph

Two digraphs are **isomorphic** if their underlying graphs are isomorphic

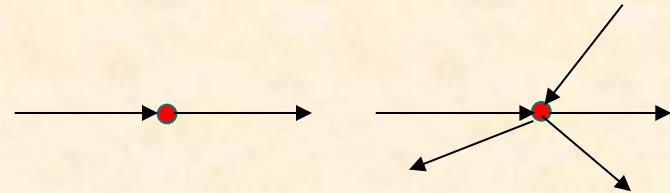
Degrees

In-degree: d_{in} 

Out-degree: d_{out} 

Total degree: $d = d_{in} + d_{out}$

Balanced, if $d_{in} = d_{out}$



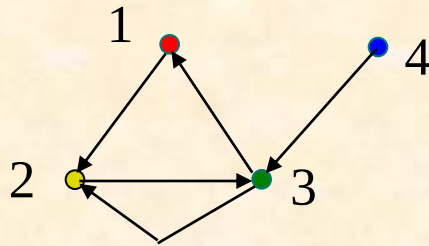
The following can be similarly defined:

- Directed walk
- Directed trail
- Directed path
- Directed circuit
- Directed tree
- Directed Eulerian graph
- Directed Hamiltonian graph

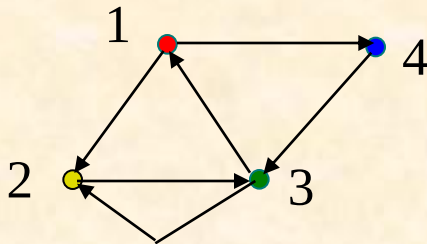
Connectivity

A digraph is **connected** if its underlying graph is connected.

A digraph is **strongly connected** if there is a directed path from any node to any other node.



← connected but not strongly connected

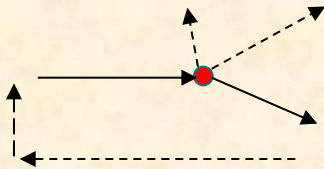


← strongly connected

Some Results

Theorem: Let G be a connected digraph. If all its nodes have $d_{in} = 1$, regardless of their d_{out} , then G has one and only one directed circuit.

Proof.

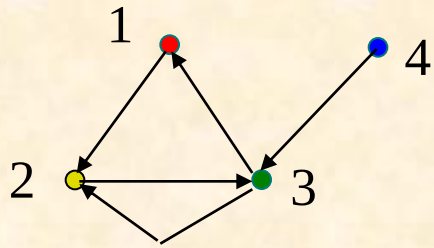


Corollary: Let G be a connected digraph. If all its nodes have $d_{out} = 1$, regardless of their d_{in} , then G has one and only one directed circuit.

Theorem: A connected (sub)digraph is a directed circuit if and only if all its nodes satisfy $d_{in} = d_{out}$.

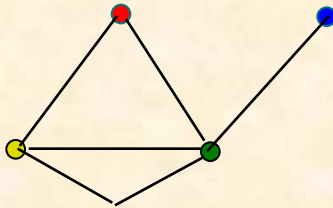
Corollary: A connected digraph contains a directed circuit if and only if it has a subgraph with all nodes satisfying $d_{in} = d_{out}$.

Adjacency Matrix



$$A_d = [a_{ij}] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (\text{asymmetric})$$

a_{ij} -- from i to j



Underlying network
A is not well-defined
due to multiple edges

More Results

Theorem: A connected digraph is directed Eulerian if and only if every node satisfies

$$d_{in} = d_{out}$$

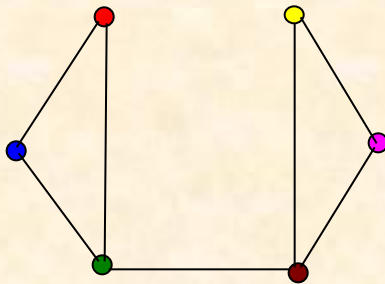
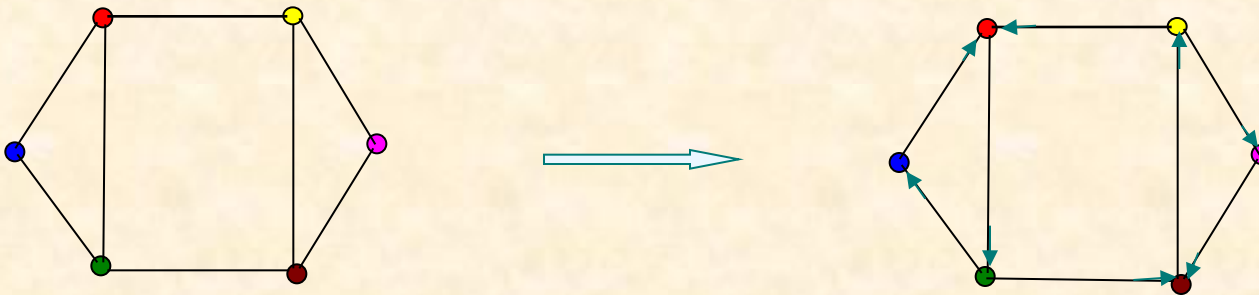
Theorem: Let G be a strongly connected digraph with N nodes. If all nodes satisfy both

$$d_{in} \geq N/2 \quad \text{and} \quad d_{out} \geq N/2$$

Then G is directed Hamiltonian.

Orientable Graphs

A graph is **orientable** if its edges can be directed such that the resulting digraph is strongly connected.



← not orientable

→ importance of bridges !

→ importance of circuits !

Some More Results

Theorem: A connected graph is orientable if and only if each edge of the graph is contained in a circuit.

Proof.

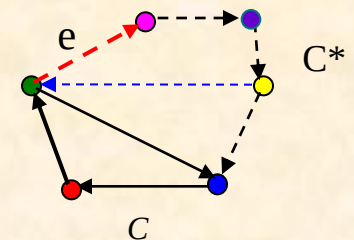
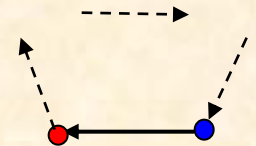
Necessity - G is orientable $\rightarrow$ every edge can be directed $\rightarrow$

Sufficiency - Choose any circuit C and direct it cyclically.

If $C = G$, done.

If $C \subset G$, then consider any adjacent edge e . By assumption, e is in some circuit $e \in C^* \subset G$. Since C^* is a circuit, there is a path to come back to e . One can then direct this path cyclically.

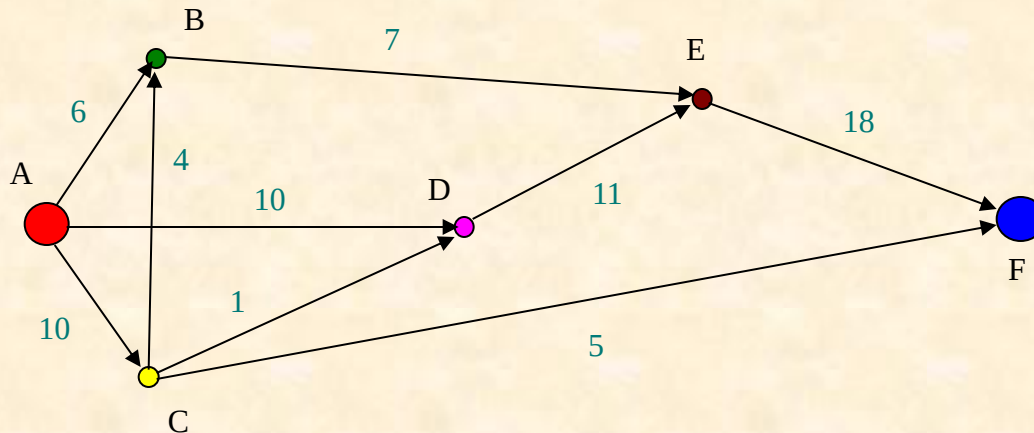
Repeat the above for all edges, then done.



Application: Maximum Flows

Consider the following roadmap, where weights are profit values.

Starting from A , follow the directed road to traverse to F , not allowed to repeat any road, finally arrive at F while making the maximum profits.



Algorithm

$$l(A) = 0$$

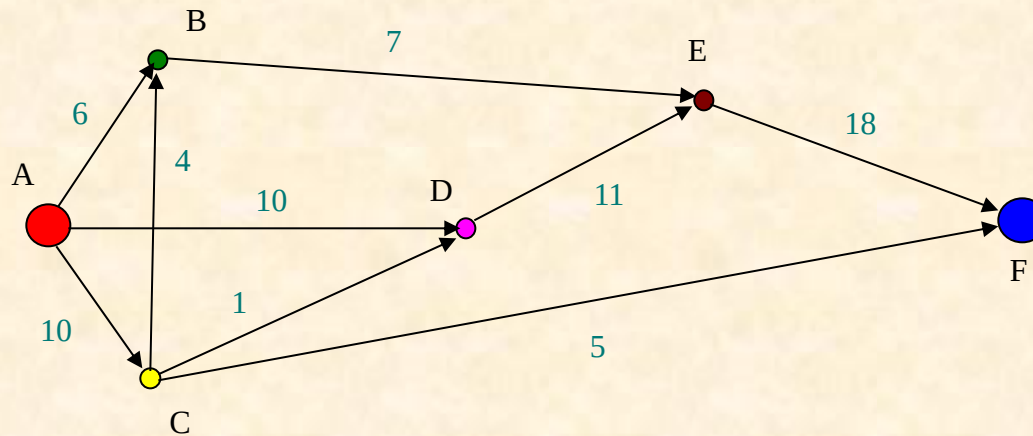
$$l(B) = l(A) + 6 = 6$$

$$l(C) = l(A) + 10 = 10 \rightarrow l(B) = l(C) + 4 = 14 \rightarrow l(E) = l(B) + 7 = 21 \rightarrow l(F) = l(E) + 18 = \mathbf{39}$$

$$l(D) = l(C) + 1 = 11 \rightarrow l(E) = l(D) + 11 = 22 \rightarrow l(F) = l(E) + 18 = \mathbf{40}$$

$$l(F) = l(C) + 5 = \mathbf{15}$$

$$l(D) = l(A) + 10 = 10 \rightarrow l(E) = l(D) + 11 = 21 \rightarrow l(F) = l(E) + 18 = \mathbf{39}$$



Optimal Solution

$$l(A) = 0$$

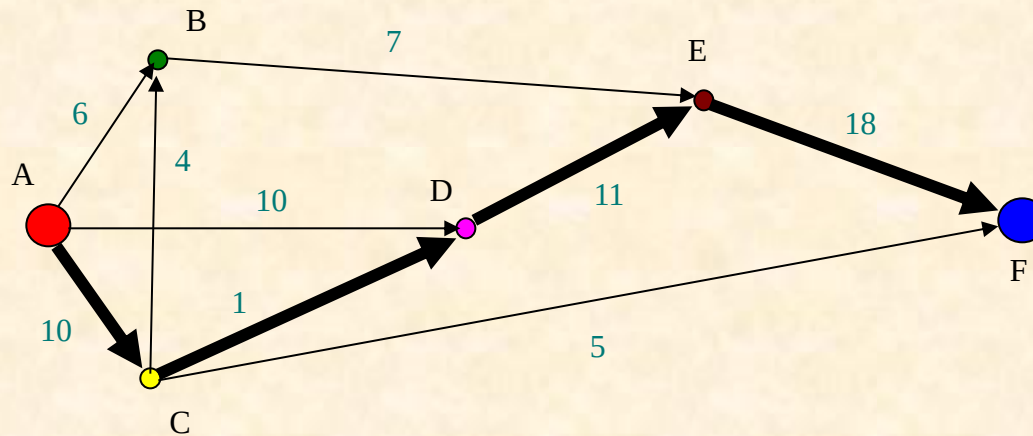
$$l(B) = l(A) + 6 = 6$$

$$l(C) = l(A) + 10 = 10 \rightarrow l(B) = l(C) + 4 = 14 \rightarrow l(E) = l(B) + 7 = 21 \rightarrow l(F) = l(E) + 18 = \mathbf{39}$$

$$l(D) = l(C) + 1 = 11 \rightarrow l(E) = l(D) + 11 = 22 \rightarrow l(F) = l(E) + 18 = \mathbf{40}$$

$$l(F) = l(C) + 5 = \mathbf{15}$$

$$l(D) = l(A) + 10 = 10 \rightarrow l(E) = l(D) + 11 = 21 \rightarrow l(F) = l(E) + 18 = \mathbf{39}$$

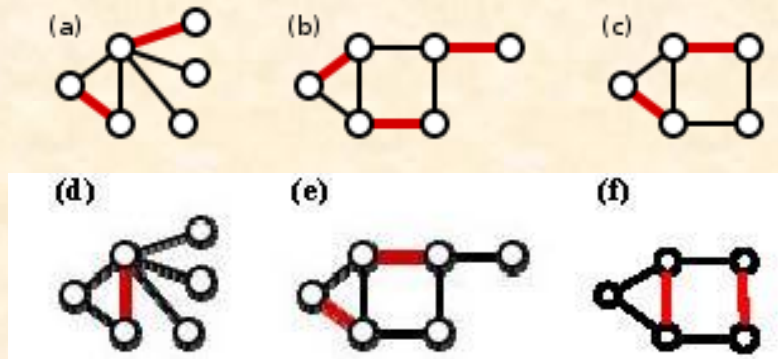


Matching

Undirected Graphs

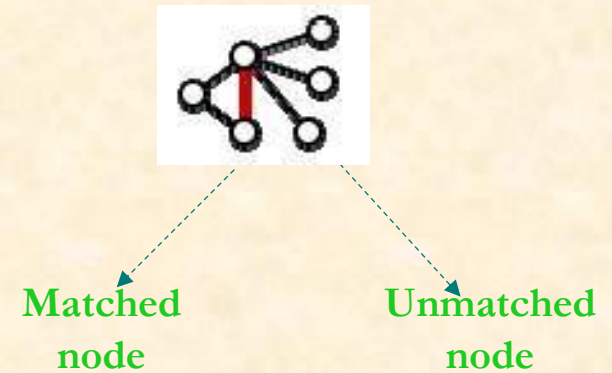
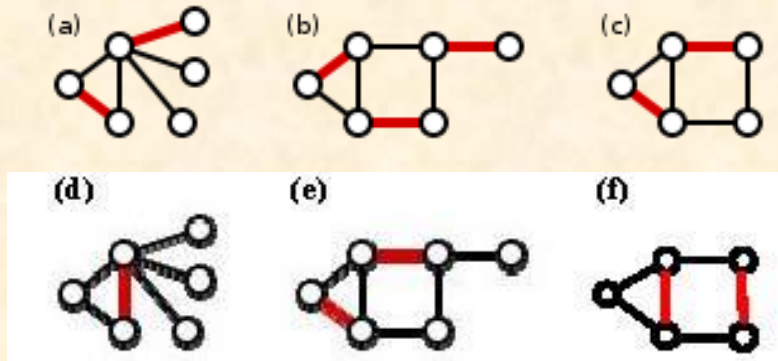
A matching M in a graph G is a set of pair-wise non-adjacent edges.

Examples: all sets of red-colored edges in the following figures are matchings:



A maximal matching is one on which no more edge can be added. For example, matchings in figures (a), (b), (c) and (f), are maximal, but those in (d) and (e) are not.

A maximum matching, furthermore, is a maximal matching that contains the largest possible number of edges.



Both **maximal** and **maximum** matchings in a graph **may not be unique**. This is clear by inspecting the above figures.

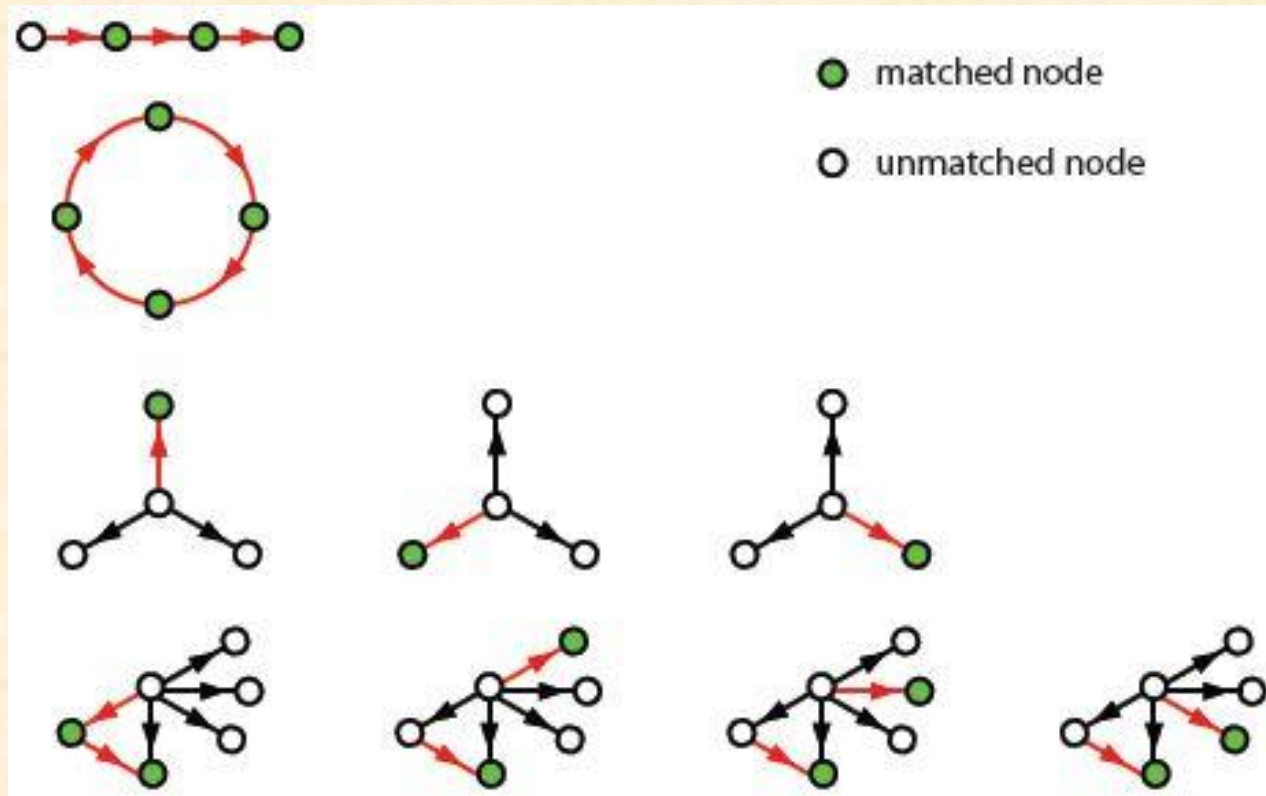
A **node** is **matched** if it is an end-point of an edge in the matching; otherwise, the node is **unmatched**.

A **perfect matching** is a matching which matches all nodes of the graph. Figure (b) above is an example of a perfect matching, while the others are not.

Every perfect matching is maximum and hence maximal.

Matching in Directed Graphs

- **Matching:** a set of directed edges without common heads and tails
- **Matched node:** the tail node of a matching edge

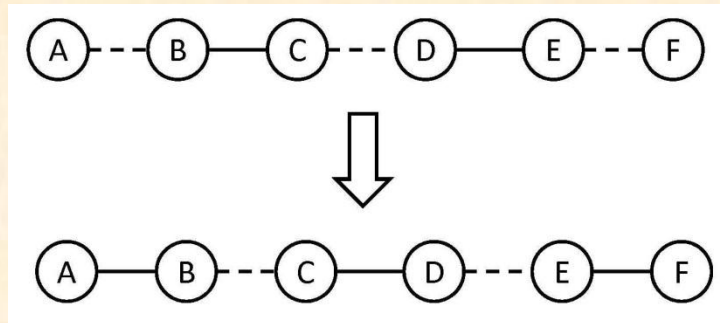


Matching Algorithm

How to find a perfect matching?

Alternating path is a path whose edges are alternately in and out of the matching (for directed graphs, also change its direction to opposite)

Example:



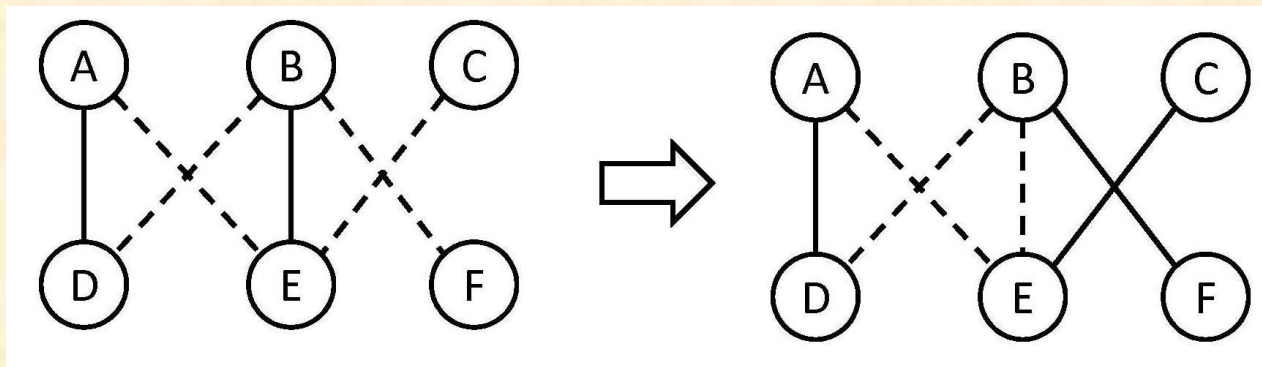
At top, there is a matching {BC, DE}, where nodes A, F are free

At bottom, it is an **alternating path**, where nodes A, F are matched

Augmenting path is an alternating path between two unmatched nodes (**free nodes**)

Example:

Consider the graph on the left-hand side, which has a matching $\{AD, BE\}$. For path C-E-B-F, its **augmenting path** is the one on the right-hand side, in which dashed lines became solid lines and solid lines became dashed lines.

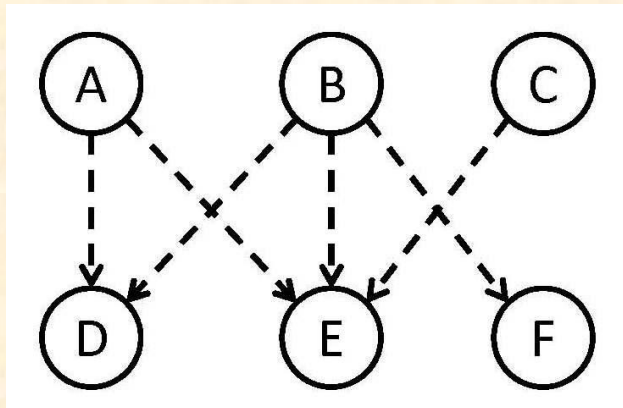


Augmenting path algorithm (to find perfect matching):

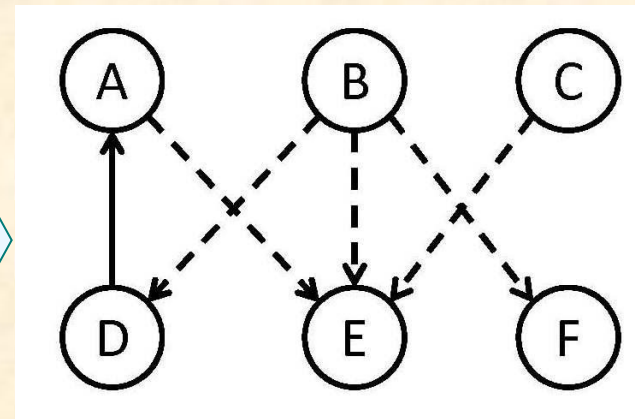
Start with an empty matching. If there is an augmenting path, do an augmentation.

Procedure: Consider the directed bipartite graph (a)

Step 1) Start from node A, search for a path which can only go to D (or E) because of the given directions; then augment it, which yields graph (b)

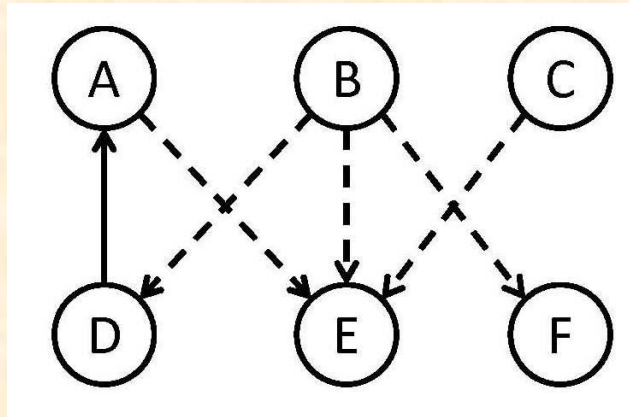


(a)

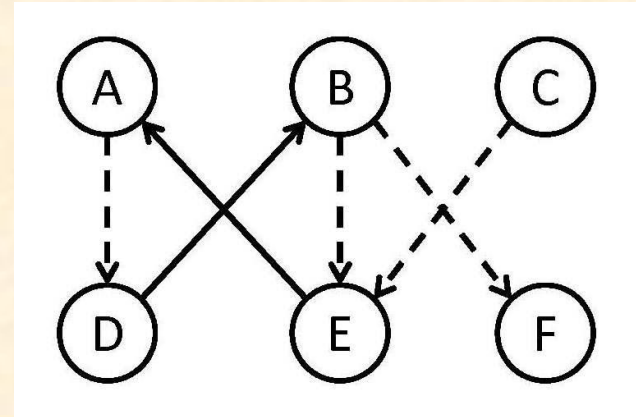
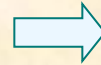


(b)

Step 2) Start from node B, search for a path which can go through B-D-A-E (or only stop at E or F); augment it, which yields graph (c)

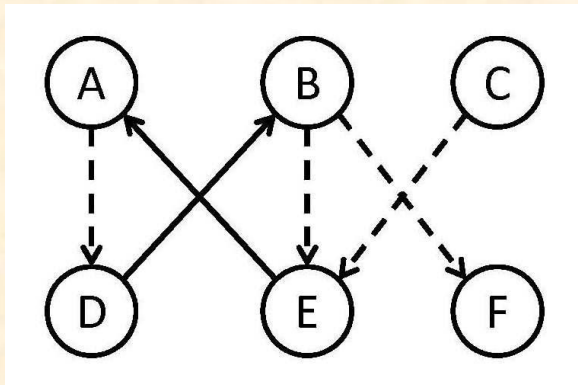


(b)

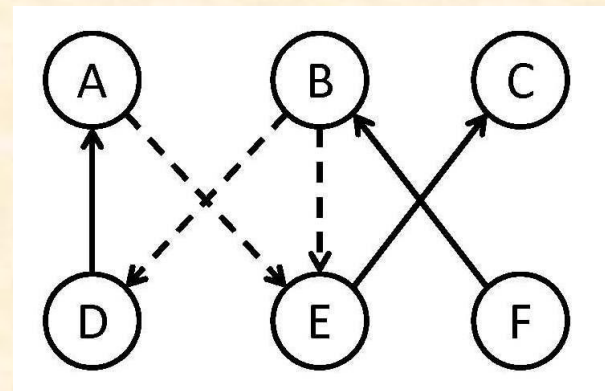
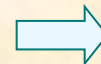


(c)

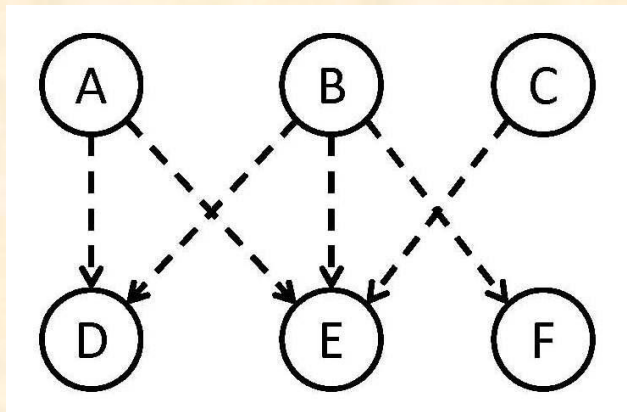
Step 3) Start from node C, search for a path which can only go to node C-E-A-D-B-F; then augment it, so as to obtain graph (d).



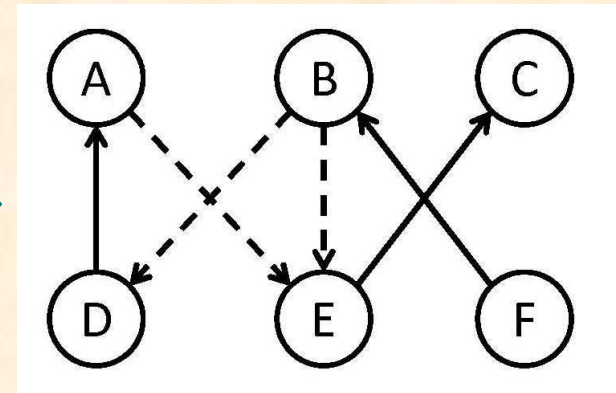
(c)



(d)



(a)



(d)

The search has completed on one side (top) of the bipartite graph, **stop**

The **final result** obtained as shown in (d) is a **solution**: one **perfect matching** of the given graph (after changing the edge directions back)

Note, again, that this solution is **not unique**.

Theorem: A matching is maximum if and only if there is no augmenting path.

End

