



IE 680 – Special Topics in Production Systems: Networks, Routing and Logistics*

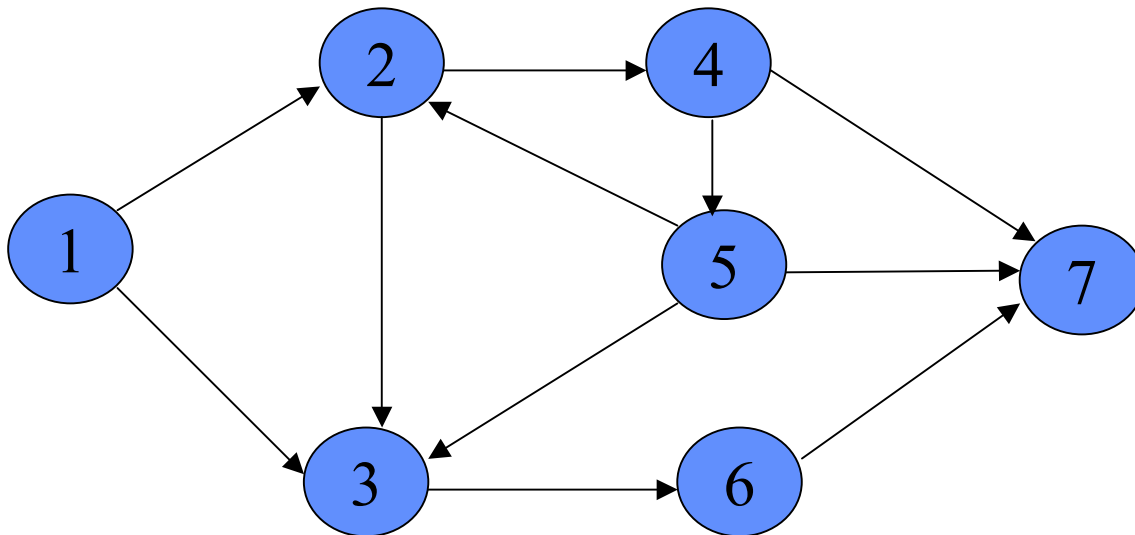
Rakesh Nagi
Department of Industrial Engineering
University at Buffalo (SUNY)

**Lecture notes from Network Flows by Ahuja, Magnanti and Wong (1993)*

UB GRAPHS, PATHS, TREES AND CYCLES

Notations and Definitions:

A graph $G=(N,A)$ consists of a set N of nodes and a set A of arcs, whose elements are ordered pairs of distinct nodes.





GRAPHS, PATHS, TREES AND CYCLES

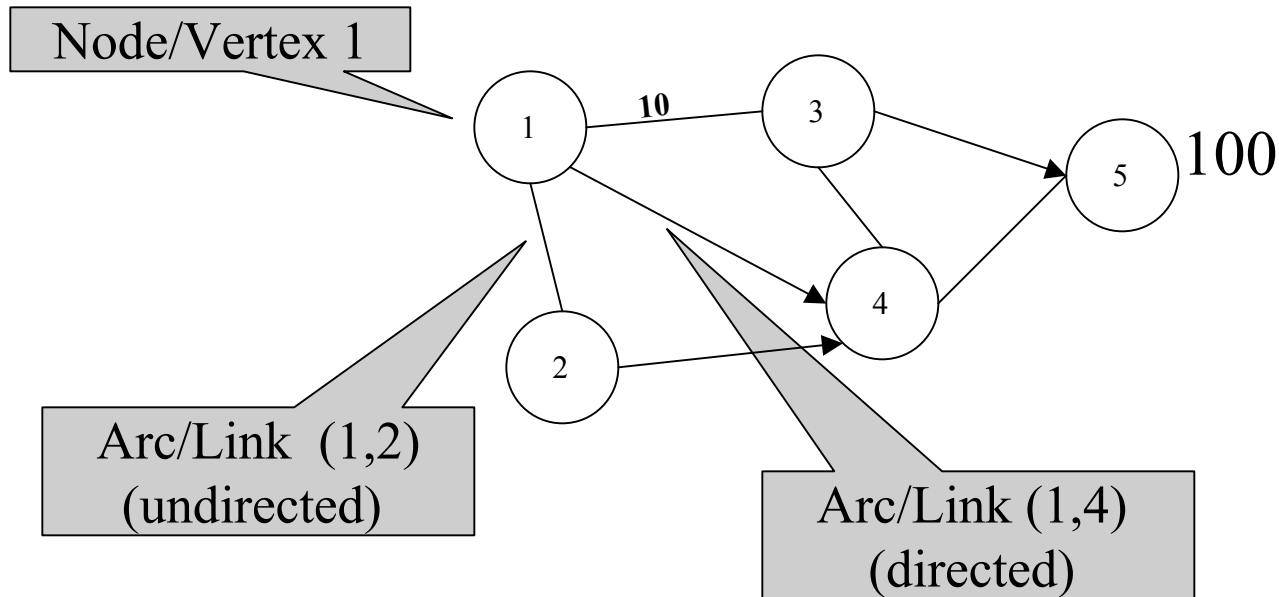
Graphs or Networks can be Directed (if all arcs are directed), Undirected (if none of the arcs are directed) or Mixed (if some of the arcs are directed).

Directed Graph: A directed network whose Nodes and/or arcs have associated numerical values (such as costs, capacities, supplies, demands, etc.)

Undirected Graph: An undirected network is one in which arcs are unordered pairs of distinct Nodes.



GRAPHS, PATHS, TREES AND CYCLES

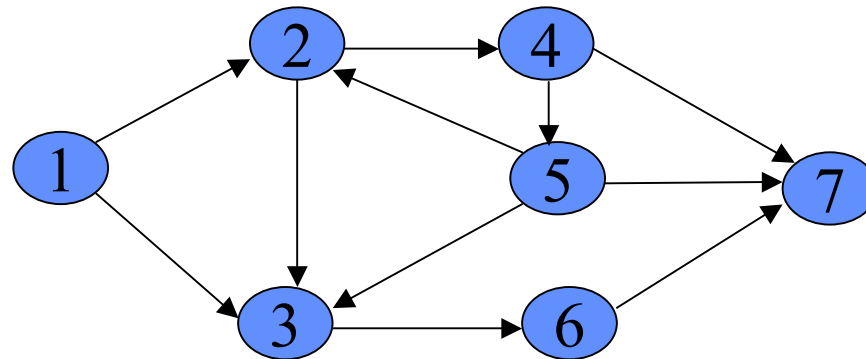


- A graph is called:
- Undirected, if all its arcs are undirected
- Directed , if all its arcs are directed
- Mixed if some arcs are directed and some are not.



GRAPHS, PATHS, TREES AND CYCLES

Example of a directed graph.



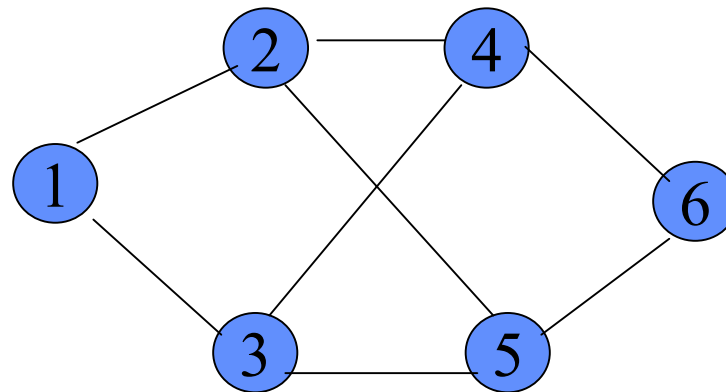
$$N = \{1, 2, 3, 4, 5, 6, 7\}$$

$$A = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 6), (4, 5), (4, 7), (5, 2), (5, 3), (5, 7), (6, 7)\}$$



GRAPHS, PATHS, TREES AND CYCLES

Example of an undirected graph.



In an undirected graph we can refer to an arc joining node pair i and j as either (i,j) or (j,i) .



GRAPHS, PATHS, TREES AND CYCLES

Basic definitions:

A directed arc (i,j) has two end points i and j . Node i is called **tail** and node j is called **head** of arc (i,j) . It **emanates** from i and **terminates** at j .

Arc (i,j) is incident to nodes i and j and is an **outgoing** arc of node i and an **incoming** arc of node j .

Node j is adjacent to node i .



GRAPHS, PATHS, TREES AND CYCLES

Degrees: The in-degree of a node is the number of incoming arcs of that node and its out-degree is the number of its outgoing arcs.

Therefore,

Degree of a node = In degree + out degree.

Sum of in degrees of all nodes = Sum of out degrees of all nodes = No. of arcs in the network.



GRAPHS, PATHS, TREES AND CYCLES

Adjacency List:

Arc adjacency list: The arc adjacency list $A(i)$ of a node i is the set of arcs emanating from that node.

Node adjacency list: The node adjacency list $A(i)$ of a node i is the set of nodes adjacent to that node.

If $|A(i)|$ represents the outdegree of node i , and 'm' represents the no. of arcs in the graph, then,

$$\sum_{i \in N} |A(i)| = m$$



GRAPHS, PATHS, TREES AND CYCLES

Subgraph:

A graph $G'=(N',A')$ is a sub graph of $G=(N,A)$, if $N' \subseteq N$ and $A' \subseteq A$.

Spanning subgraph:

A graph $G'=(N',A')$ is a spanning subgraph of $G=(N,A)$, if $N'=N$ and $A' \subseteq A$.



GRAPHS, PATHS, TREES AND CYCLES

Walk:

Is a sequence of arcs (or nodes) without the explicit mention of the nodes (without explicit mention of the arcs).

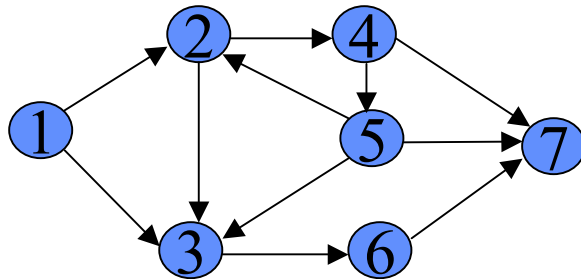
Directed walk:

A directed walk is an “oriented” version of a walk (Walk 2 on slide 11).

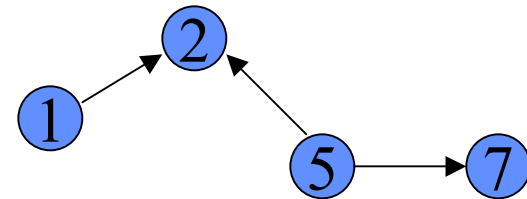


GRAPHS, PATHS, TREES AND CYCLES

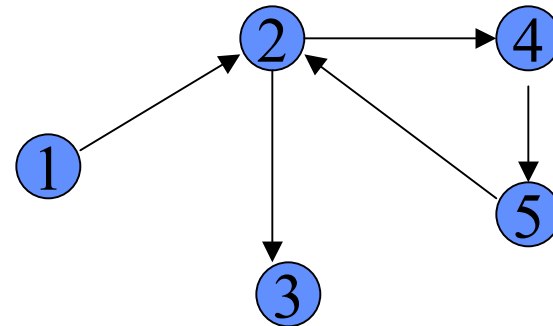
Example for walks:



Original network



Walk 1: 1-2-5-7



Walk 2: 1-2-4-5-2-3



GRAPHS, PATHS, TREES AND CYCLES

Path:

A (an elementary) path is a walk without any repetition of nodes.

An arc(i,j) in the path is a **forward arc** if the path visits node i prior to visiting node j and is an **backward arc** otherwise.

Directed path :

A directed path is a directed walk without repetition of nodes. In other words, a directed path has **no backward arcs**. Can be stored using $pred(j) = i$ operator (for directed arc (i,j)).

Note: Some authors refer to a simple path, which does not have the same arc more than once.

Obviously, an elementary path is also simple, but not necessarily vice-versa.

Chains: Non-directed counterpart of the path

GRAPHS, PATHS, TREES AND CYCLES

Loop:

Is an arc whose head node is the same at the tail node.

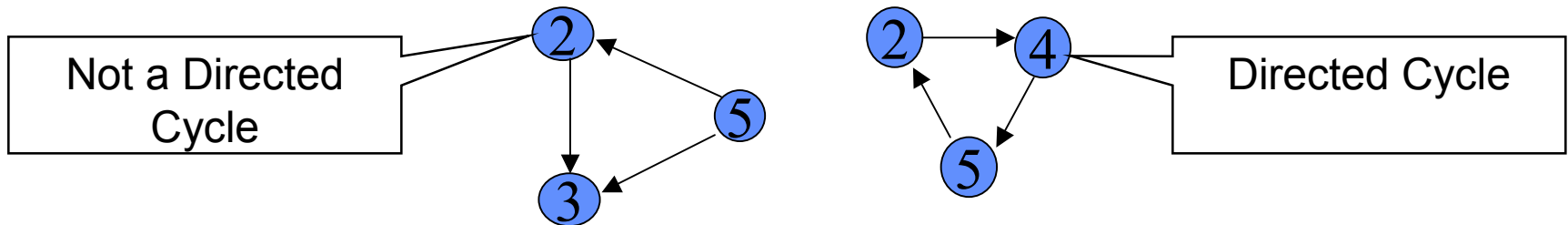
Cycle:

A cycle is a path $i_1-i_2-\dots-i_r$ together with the arc (i_r-i_1) or (i_1-i_r) .

A cycle can be referred as $i_1-i_2-i_3-\dots-i_r-i_1$.

Directed Cycle (Circuit):

Is a directed path $i_1-i_2-\dots-i_r$ together with the arc (i_r, i_1)



Acyclic Graph: A graph is acyclic if it has no directed cycles.

Connectivity: A graph is connected if every pair of its nodes is connected; otherwise it is disconnected.

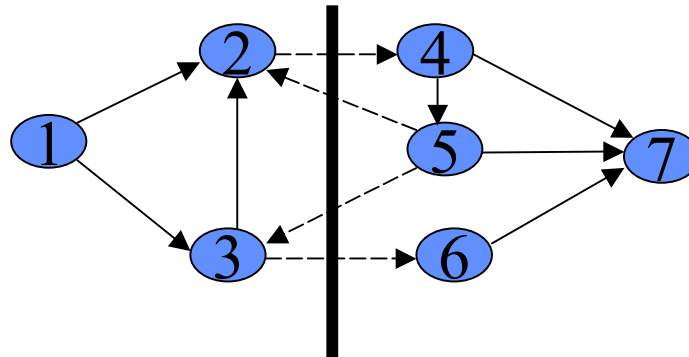


GRAPHS, PATHS, TREES AND CYCLES

Cut $[S, \hat{S}]$:

Is a partition of the node set N into two parts, S and $\hat{S} = N - S$.
Each Cut defines a set of arcs consisting of those arcs that have one endpoint in S and the other in \hat{S} .

$S = \{1, 2, 3\}$ and $\hat{S} = \{4, 5, 6, 7\}$ and the set of arcs in the cut are $\{(2, 4), (5, 2), (5, 3), (3, 6)\}$.





GRAPHS, PATHS, TREES AND CYCLES

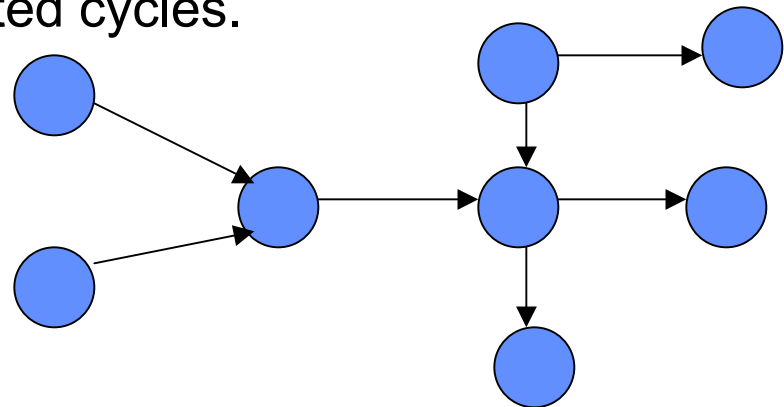
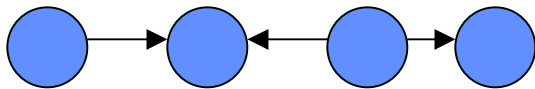
s-t Cut:

Is defined w.r.t. two distinguished nodes s and t and is a cut $[S, \check{S}]$ satisfying the property that $s \in S$ and $t \in \check{S}$.

From the above graph, if $s=1$ and $t=6$ then the cut is an s-t cut; but if $s=1$ and $t=3$ it is not an s-t cut.

Tree:

Is a graph that has no connected cycles.





GRAPHS, PATHS, TREES AND CYCLES

Properties of trees:

- a) A tree with n nodes has exactly $n-1$ arcs.
- b) A tree has at least two leaf nodes (i.e. nodes with degree 1).
- c) Every two nodes of a tree are connected by a unique path.

Sub-tree:

A connected subgraph of a tree is called a subtree.



GRAPHS, PATHS, TREES AND CYCLES

- Other definitions of non-directed trees
 1. A connected graph of n vertices and $(n-1)$ links
 2. A connected graph without a cycle
 3. A graph in which every pair of vertices is connected with one and only one elementary path



Rooted tree:

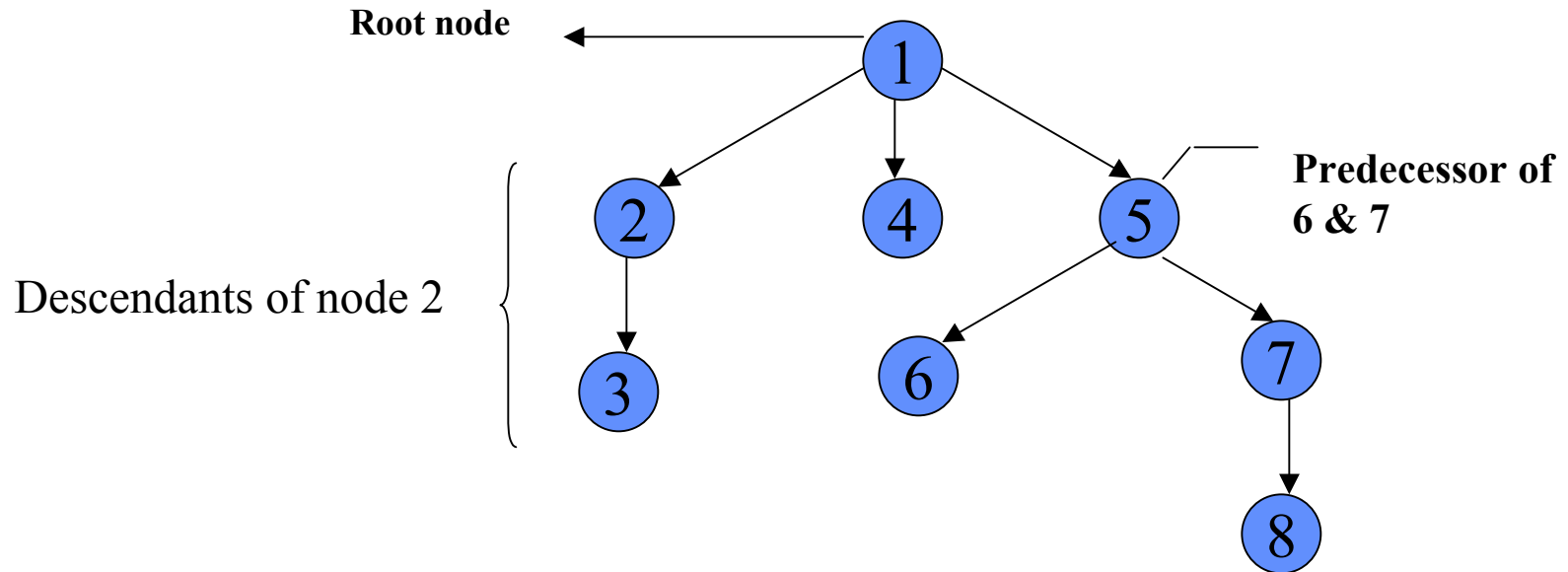
A rooted tree is a tree with a specially designated node, called as its root.

Note:

1. Each node i (except the root node) has an unique predecessor, which is the next node on the unique path in the path from the node to the root.
2. The descendants of a node consists of the node itself, its successors, its successors and so on.
3. A node having descendants is called the ancestor of all its descendants.



GRAPHS, PATHS, TREES AND CYCLES



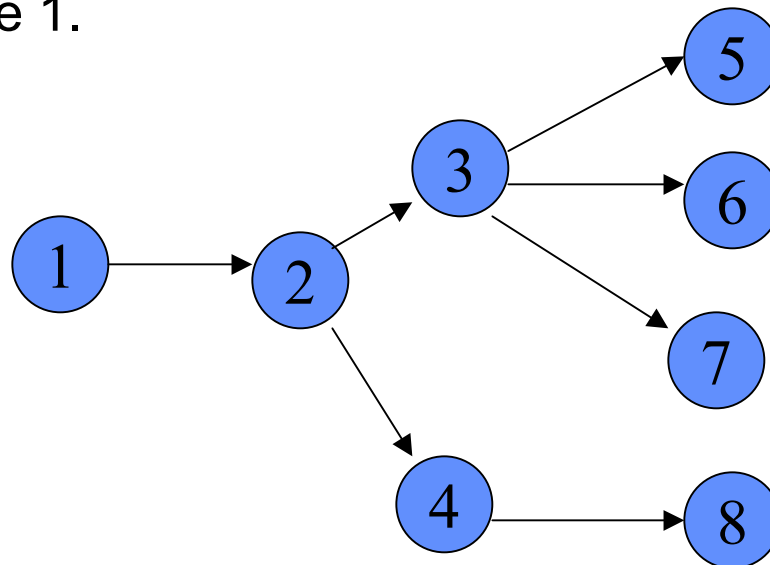


GRAPHS, PATHS, TREES AND CYCLES

Directed -out tree:

is a tree rooted at node s if the unique path in the tree from node s to every other node is a directed path.

Note: Every node in the directed-out tree (except root node) has indegree 1.



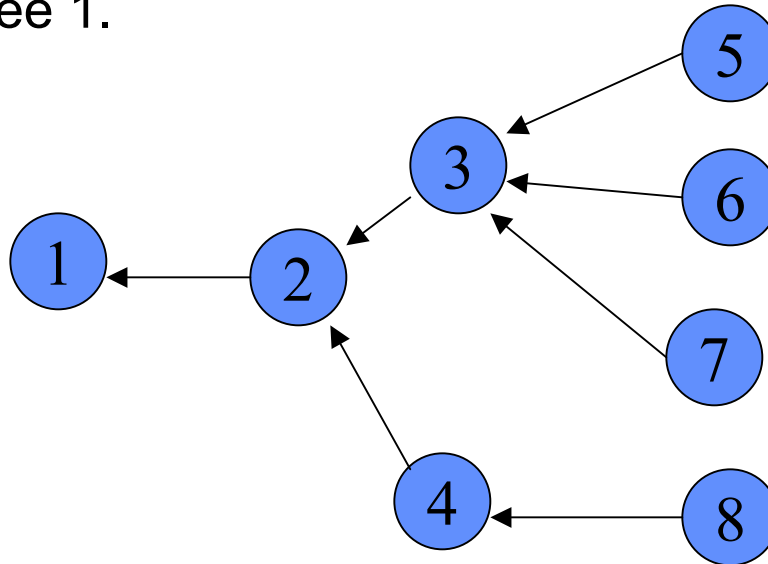


GRAPHS, PATHS, TREES AND CYCLES

Directed -in tree:

is a tree rooted at node s if the unique path in the tree from any node to node s is a directed path.

Note: Every node in the directed-in tree (except root node) has outdegree 1.

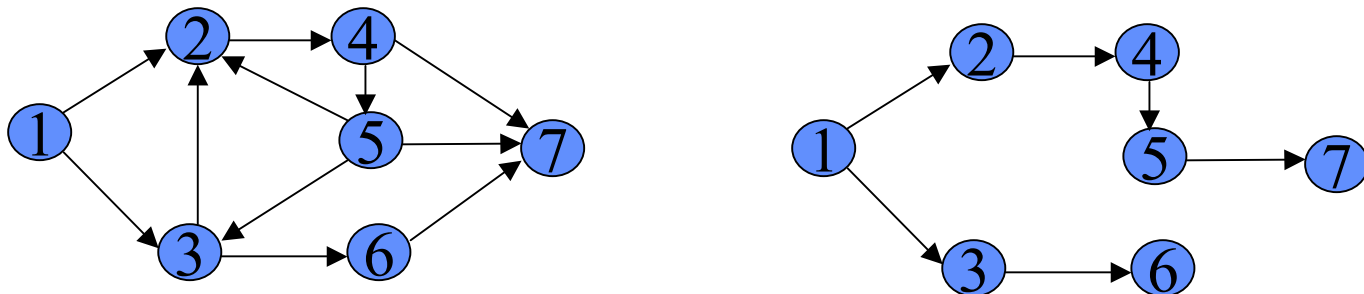




GRAPHS, PATHS, TREES AND CYCLES

- Spanning tree:

A tree T is a spanning tree of G if T is a spanning subgraph of G . Every spanning tree of a connected n -node graph G has $(n-1)$ arcs.



The arcs belonging to a spanning tree are called tree arcs and the rest are called nontree arcs.



GRAPHS, PATHS, TREES AND CYCLES

- Minimal Spanning tree algorithm:

The algorithm deals with linking all the nodes of a network directly or indirectly, using the shortest length of connecting branches.

Applications: Creation of a network of paved roads between several rural towns, where the objective is to minimize the total miles of the paved roads.

- Algorithm:

Step 1: Select all the vertices.

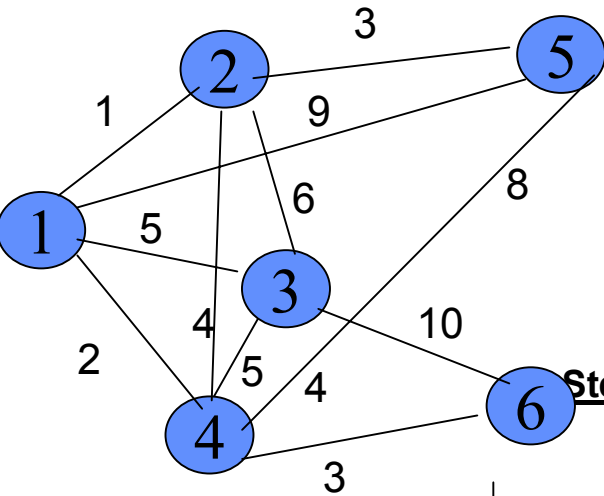
Step 2: Choose the arc with the smallest weight; such that addition of the arc does not create a cycle.

Step 3: Repeat step 1 until all nodes are connected.

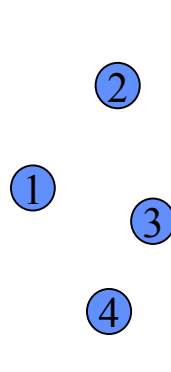


GRAPHS, PATHS, TREES AND CYCLES

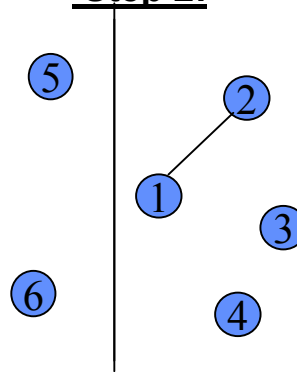
Example :



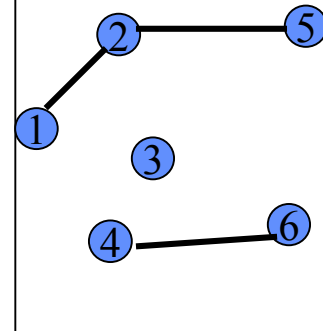
Step 1:



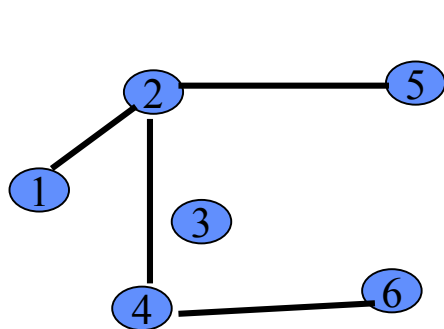
Step 2:



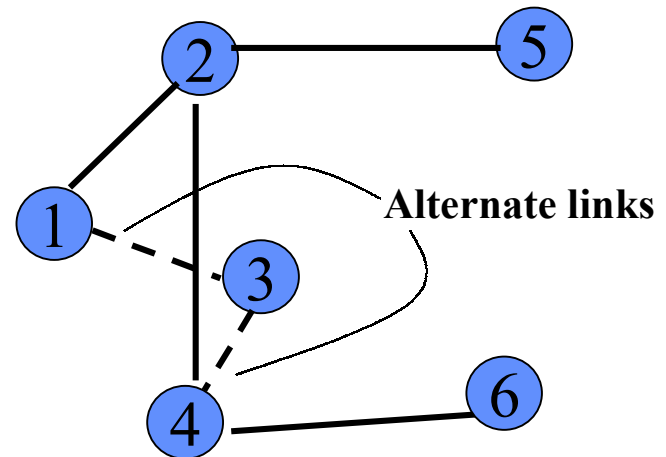
Step 3:



Step 4:



Step 5:

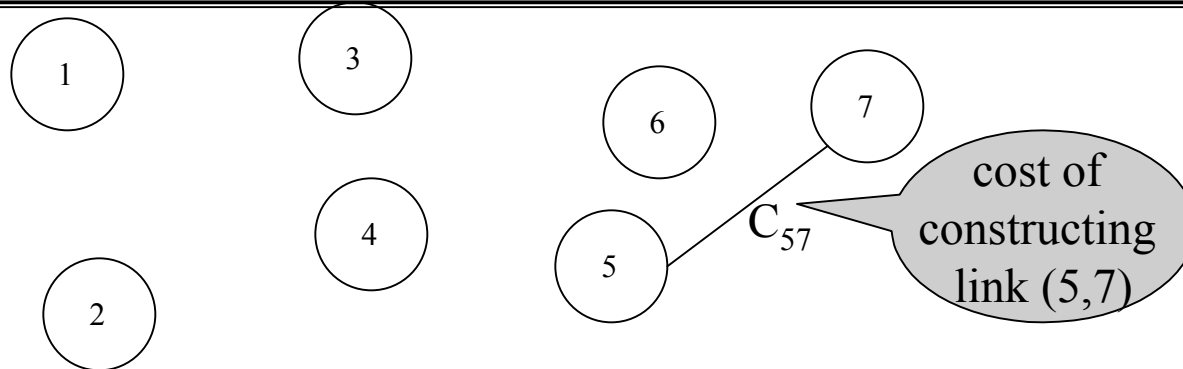


The minimal spanning tree =

$$1+3+3+4+5 = 16$$



Minimal-Spanning Tree Model (For Undirected Graphs Only)



- Let c_{ij} be the cost of constructing an arc that links nodes i and j , thereby allowing direct bi-directional travel between these two nodes.
- Given the cost of constructing all the arcs in the graph, find the cheapest set of arcs to construct, such that travel will be possible between every pair of nodes in the above graph. In other words, a transportation network that connects all the points while minimizing total distance or cost of construction.
- It can be shown that the cheapest such graph is always a *tree*, i.e, the cheapest graph will not have a cycle. Hence this is called the Minimal Spanning Tree Problem
- Applications: Network Design in Telecommunication, Transportation.

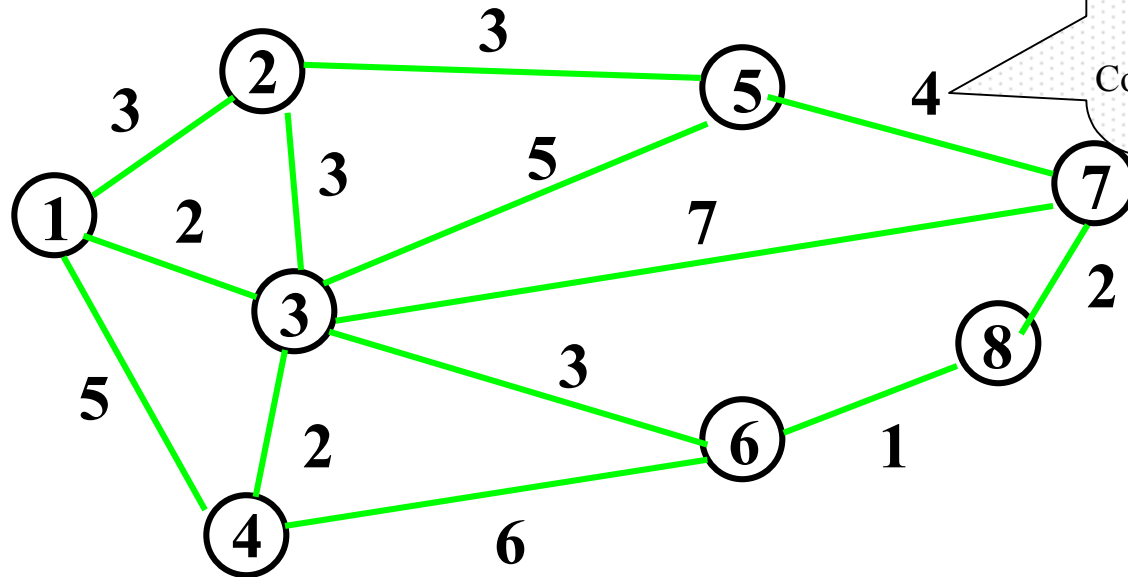


Minimal-Spanning Tree Algorithm (Prim's Algorithm)

1. Select any node in the network.
2. Connect this node to the nearest node that minimizes the total distance.
3. Considering all of the nodes that are now connected, find and connect the nearest node that is not connected.
4. Repeat the third step until all nodes are connected.
5. If there is a tie in the third step and two or more nodes that are not connected are equally near, arbitrarily select one and continue. A tie suggests that there might be more than one optimal solution.



Minimal-Spanning Tree Lauderdale Construction



Direct Distance between
nodes 6 and 7 is 4

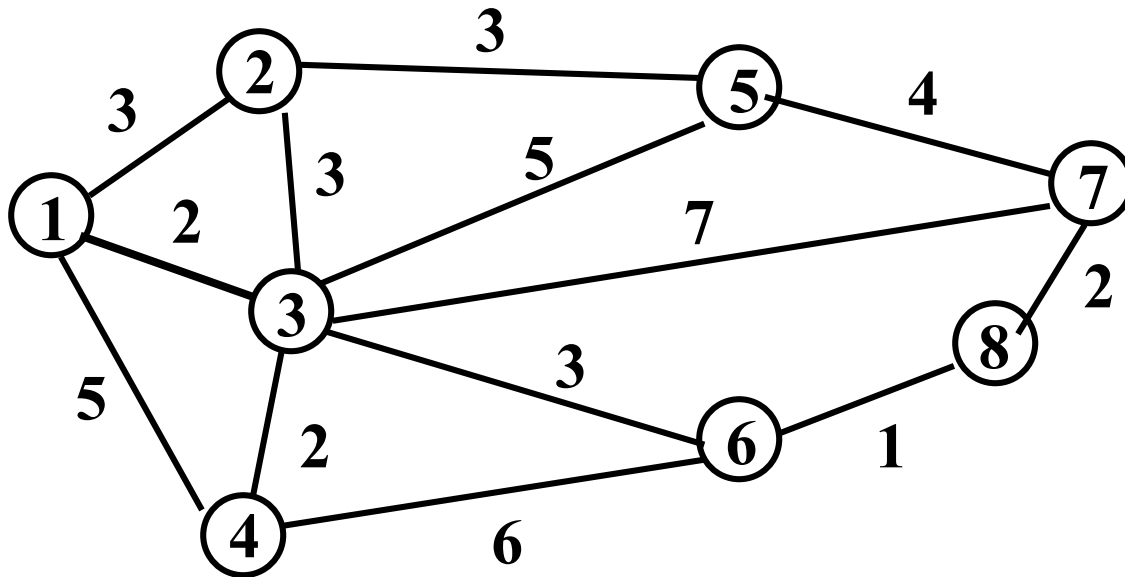
OR

Cost of constructing this
link (5,7) is 4

Check first that the graph is undirected.

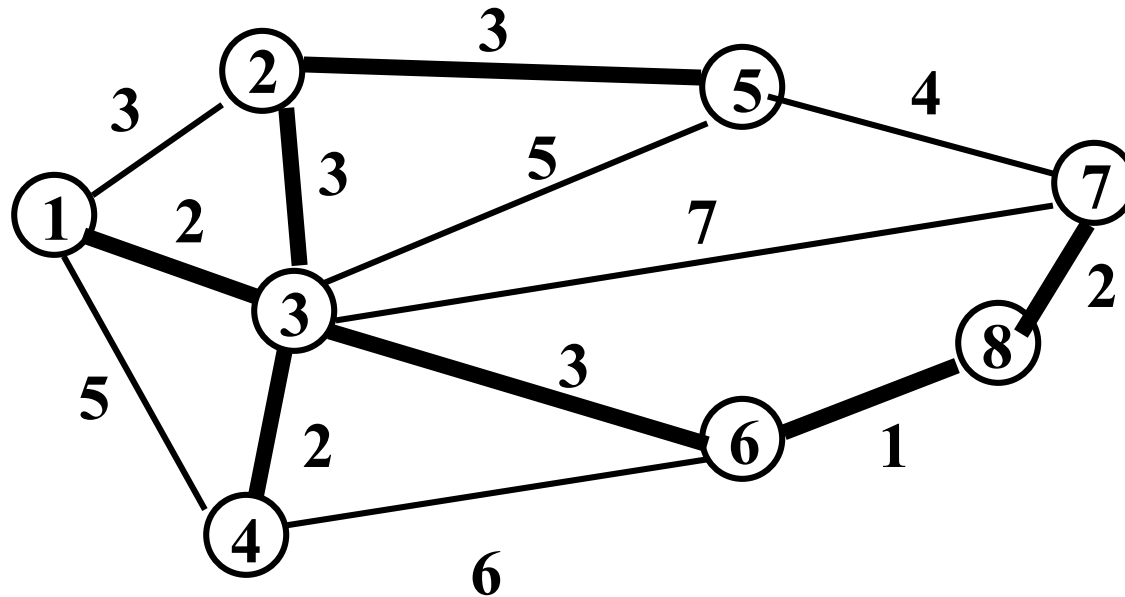


Minimal-Spanning Tree (Prim's Algorithm)





Minimal-Spanning Tree Final Result



Cheapest/Minimum Tree has a total distance or cost of 16.



Minimal Spanning Tree (Kruskal's Greedy Algorithm)

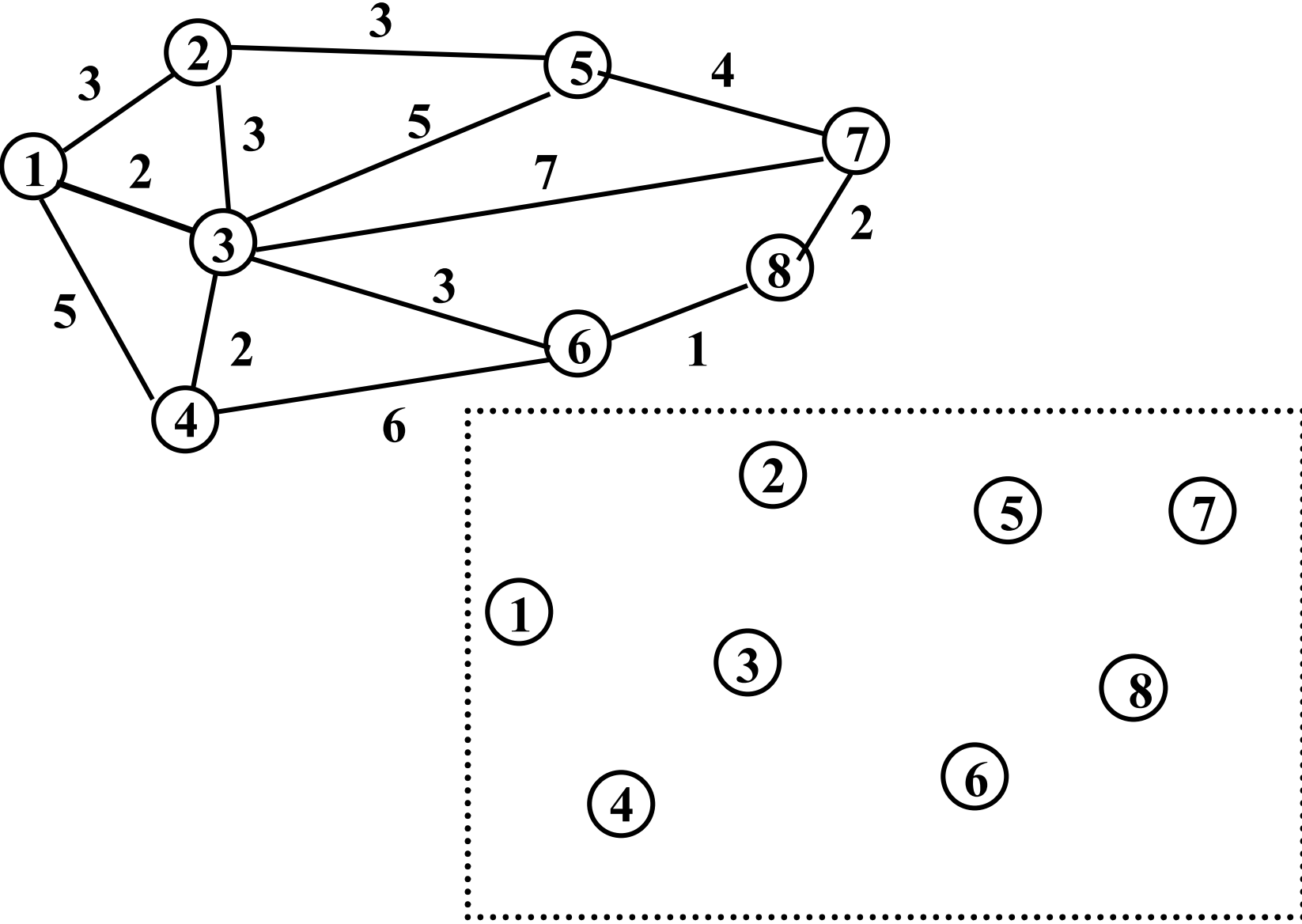
Step 1: Begin with just the nodes and none of the arcs constructed.

Step 2: Of the remaining arcs that could be constructed, choose the cheapest/smallest. Break ties arbitrarily.

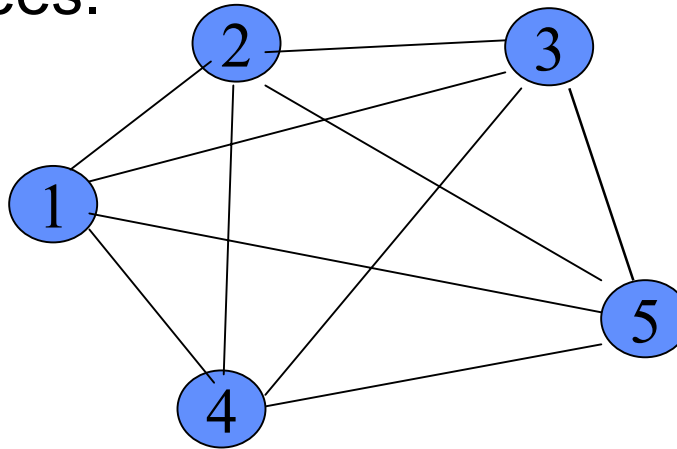
If constructing this arc does not create a cycle in the graph, construct it. Else discard this arc from further consideration.

Step 3: Repeat Step 2 until all nodes are connected.

Minimal Spanning Tree - Kruskal's Greedy Algorithm



- How many trees do we have from a graph?
 - Let $T(n)$ be the number of trees with n labeled points. In 1889, Cayley proved that $T(n) = n^{n-2}$.
 - For example, a complete graph of 5 nodes has 125 trees.



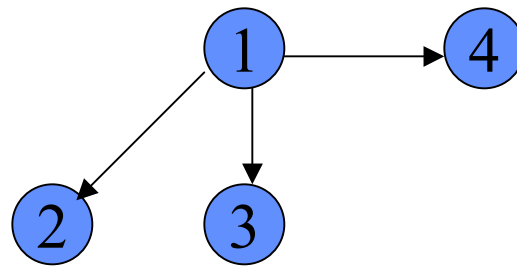
In Electrical Engineering, the number of trees in a graph indicates the complexity of the graph



GRAPHS, PATHS, TREES AND CYCLES

- Forest:

A graph that contains no cycle is a forest. Hence, a forest is a collection of trees.



- Another definition of directed trees (arborescence):

- A directed tree is a directed graph without a circuit, for which the indegree of every vertex, except one (called the root of the tree) is unity. Indegree of the root is zero.



GRAPHS, PATHS, TREES AND CYCLES

- Fundamental cycles:

If T is a spanning tree of a graph G , the addition of any nontree arc to the spanning tree T creates exactly one cycle. This is referred to as a fundamental cycle of G w.r.t T .

If m =no. of arcs, and n =no. of nodes of a network, then

No. of non-tree arcs in the network = $m - n + 1$

therefore, it has $m - n + 1$ fundamental cycles.

Note: If we delete any arc in a fundamental cycle we obtain a spanning tree.

- Fundamental Cuts:

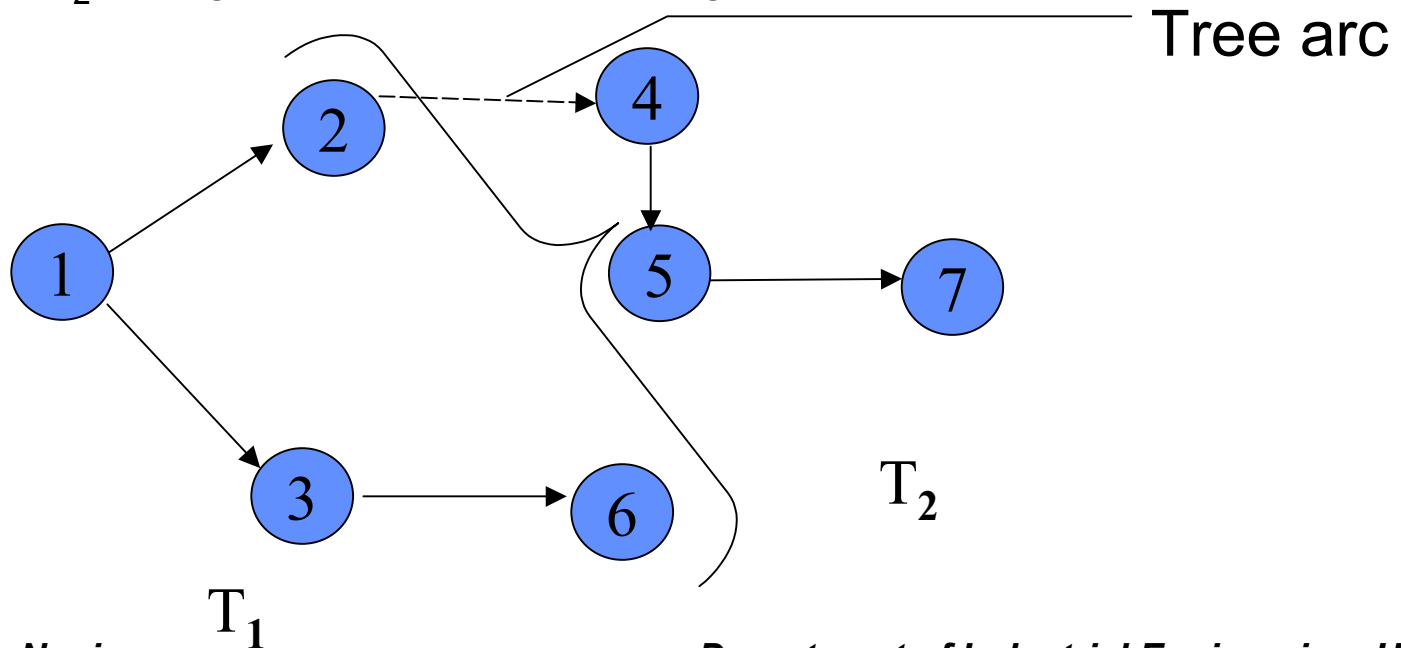
The deletion of any tree arc of a spanning tree T produces a disconnected graph containing two subtrees T_1 and T_2 . Such a cut is called a fundamental cut of G w.r.t tree T .



GRAPHS, PATHS, TREES AND CYCLES

Note:

1. Since a spanning tree contains $(n-1)$ arcs, the network has $(n-1)$ fundamental cuts w.r.t any tree.
2. When we add any arc in the fundamental cut to the two subtrees T_1 and T_2 we again obtain the spanning tree.





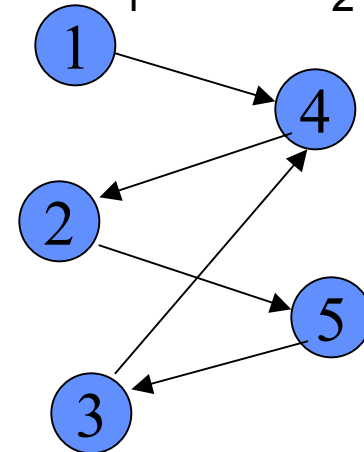
GRAPHS, PATHS, TREES AND CYCLES

- Bipartite Graph:

A graph $G=(N,A)$ is a Bipartite graph if we can partition its node set into two subsets N_1 and N_2 so that for each arc (i,j) in A either

- (i) $i \in N_1$ and $j \in N_2$; or
- (ii) $i \in N_2$ and $j \in N_1$

$N_1=\{1,2,3\}$ and $N_2=\{4,5\}$



Note: A graph G is a bipartite graph if and only if every cycle in G contains an even no. of arcs.



GRAPHS, PATHS, TREES AND CYCLES

- Network representation in a computer:

Two types of information need to be stored:

1. Network topology i.e. node and arc structure.
2. Data such as costs, capacities, demands/ supplies associated with the nodes and arcs in the network.

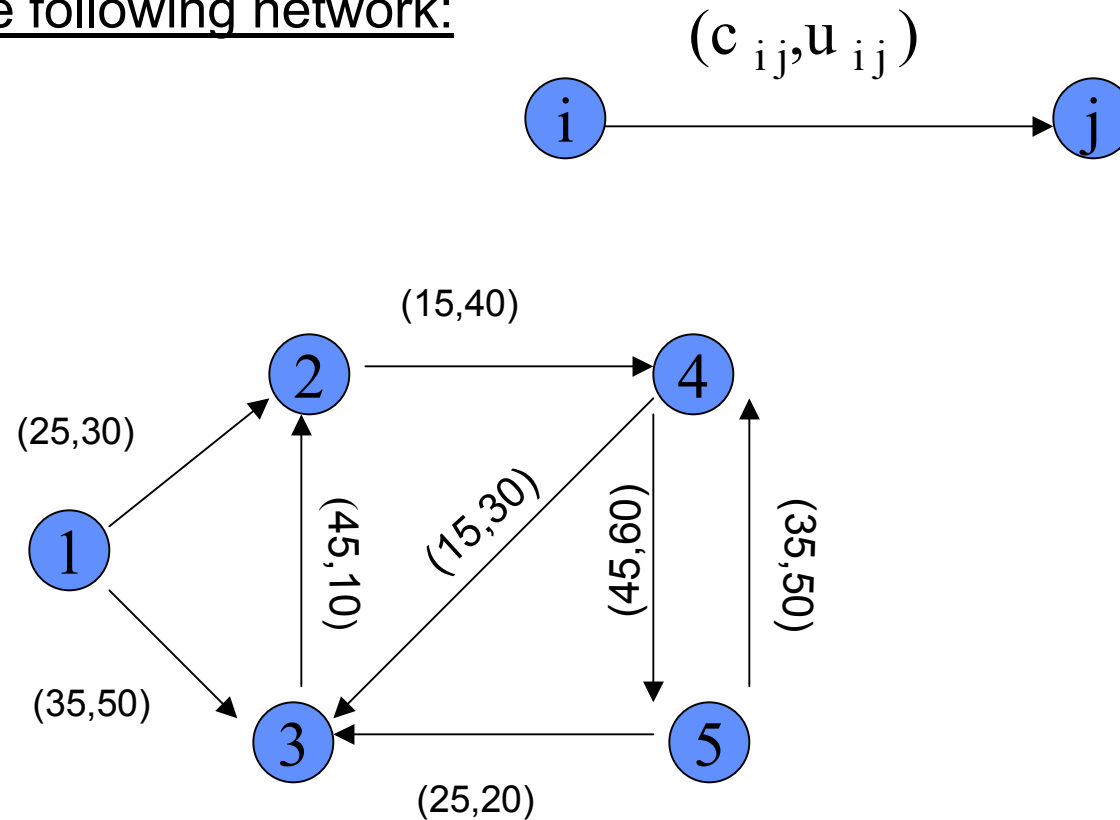
- Node-Arc incidence matrix:

This representation stores the network as an $n \times m$ matrix N , which contains one row for each node of the network and one column for each arc.



GRAPHS, PATHS, TREES AND CYCLES

Consider the following network:





GRAPHS, PATHS, TREES AND CYCLES

The column corresponding to arc (i,j) has only two non-zero elements: It has a +1 in the row corresponding to node i , and -1 in the row corresponding to node j .

	(1,2)	(1,3)	(2,4)	(3,2)	(4,3)	(4,5)	(5,3)	(5,4)
1	1	1	0	0	0	0	0	0
2	-1	0	1	-1	0	0	0	0
3	0	-1	0	1	-1	0	-1	0
4	0	0	-1	0	1	1	0	-1
5	0	0	0	0	0	-1	1	1



GRAPHS, PATHS, TREES AND CYCLES

Properties:

1. Only $2m$ out of the nm entries are non-zero, all of its non-zero entries are $+1$ or -1 .
2. Each column has exactly one $+1$ and one -1 .
3. The no. of $+1$'s in a row equals the outdegree of the corresponding node and the no. of -1 's in the row indicate the indegree of the node.

Disadvantages:

1. Since the matrix contains few non-zero elements it is not space efficient.
2. Because of inefficient storage of the network topology, it rarely produces efficient algorithms.



GRAPHS, PATHS, TREES AND CYCLES

- Node-node Adjacency matrix (or adjacency matrix representation):

The network is stored as an $n \times n$ matrix $\hat{H} = \{h_{ij}\}$.

The matrix has a node and column corresponding to each node, and its $(i, j)^{\text{th}}$ entry h_{ij} equals 1 if $(i, j) \in A$ and equals 0 otherwise.

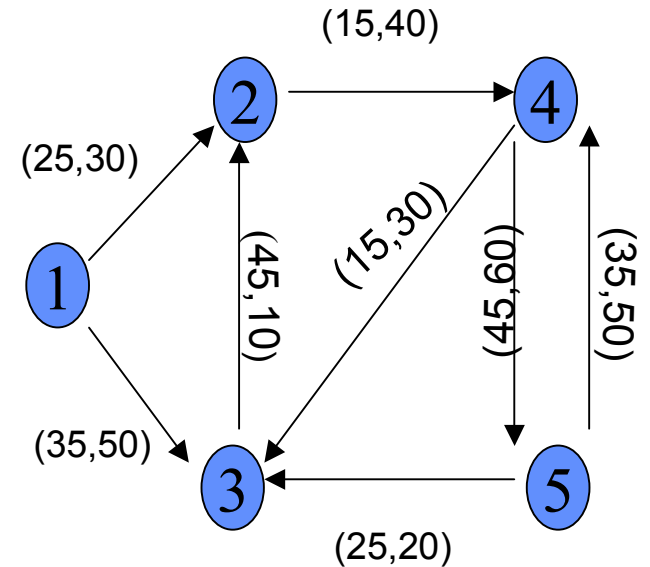
If we need to store the arc costs and capacities as well as the network topology, we can store this information in two additional $n \times n$ matrices.



GRAPHS, PATHS, TREES AND CYCLES

Example:

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	0
3	0	1	0	0	0
4	0	0	1	0	1
5	0	0	1	1	0





GRAPHS, PATHS, TREES AND CYCLES

Properties:

- 1) The adjacency matrix has only n^2 elements only m of which are non-zero.
- 2) The cost or capacity of any arc (i, j) can be found out by looking at the i th j th element in the cost or capacity matrix.
 - The indegree of any node j can be found out by scanning column j . (if the i th element of this column has a non-zero entry, (i, j) is an arc in the network.
 - The outdegree of the node i can be found out by scanning row i . (if the j th element of this row has a non-zero entry, (i, j) is an arc in the network.



GRAPHS, PATHS, TREES AND CYCLES

Adjacency lists:

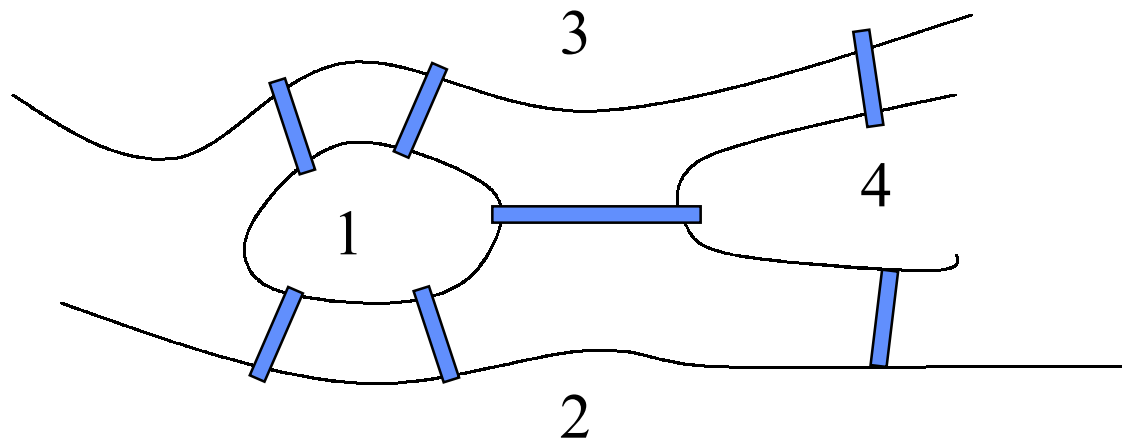
This stores the node adjacency list of each node as a singly linked list. (A linked list is a collection of cells each containing one or more lists). The cell corresponding to arc (i, j) can have many fields and each field can store information, for example for node j , arc cost or arc capacity. Each cell contains some additional field called link which stores a pointer to the next cell in the adjacency list. (the last cell in any network will have a link with value zero).





Problem of the Königsberg Bridges

- The famous problem of the Königsberg bridges was solved by the mathematician Leonhard Euler in 1736. This is regarded as the first problem in graph theory and topology. The city of Königsberg in East Prussia is located on the banks and on the two islands of the river Pregel. The various sections of the city were joined by seven bridges, as indicated in the Figure below.





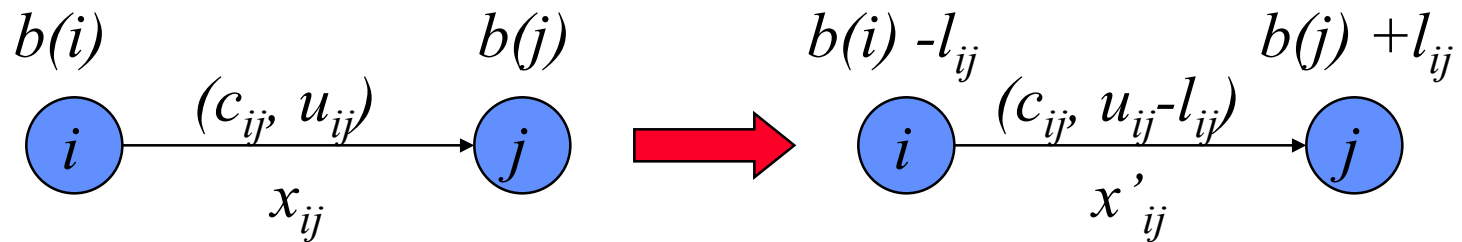
Network Transformations

- Undirected arcs \rightarrow Directed arcs
 - Undirected arc is denoted by $\{i,j\}$; Let $c_{ij} \geq 0$
 - Arc has no lower bound ($l_{ij} = 0$)
 - Replace it with two directed arcs (i,j) and (j,i)
 - In constraints: $x_{ij} + x_{ji} \leq u_{ij}$
 - In objective: $c_{ij} x_{ij} + c_{ij} x_{ji}$
 - Note that one of the x 's will be zero
 - Arc has lower bound
 - Replace it with two directed arcs (i,j) and (j,i)
 - Each with capacity u_{ij}
 - Each with cost c_{ij}
 - In constraints: $x_{ij} + x_{ji} \geq l_{ij}$



Network Transformations

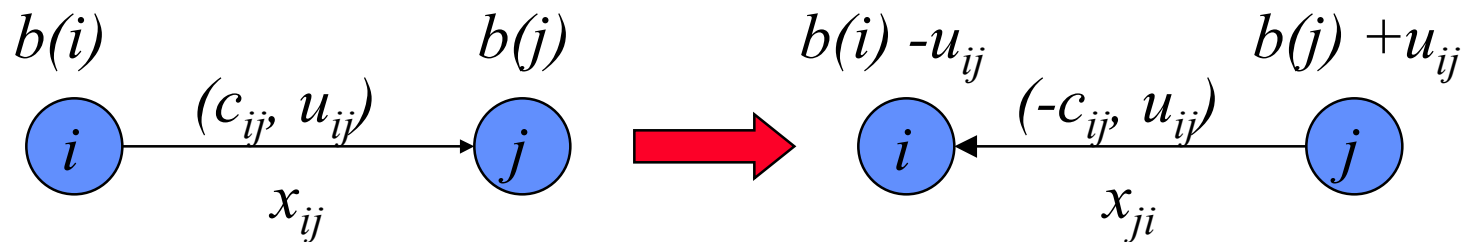
- Removing non-zero lower bounds





Network Transformations

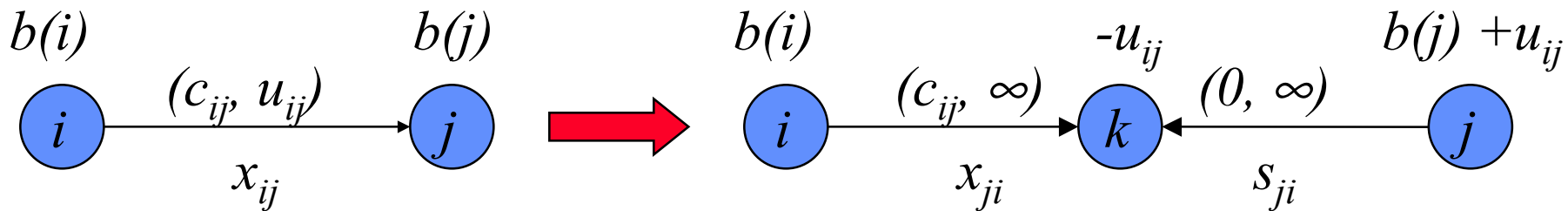
- Arc reversal





Network Transformations

- Removing arc capacities





Network Transformations

- Node splitting



Network Transformations

- Working with reduced costs



Network Transformations

- Working with residual networks