Module 8

# GRAPH

# INTRODUCTION TO GRAPHS

Definition: A simple graph $G = (V, E)$ consists of V, a nonempty set of vertices, and E, a set of unordered pairs of distinct elements of V called edges.

A simple graph is just like a directed graph, but with no specified direction of its edges.

Sometimes we want to model multiple connections between vertices, which is impossible using simple graphs.

In these cases, it have to use multigraphs.

# INTRODUCTION TO GRAPHS

Definition: A multigraph $G = (V, E)$ consists of a set $V$ of vertices, a set $E$ of edges, and a function $f$ from $E$ to $\{\{u, v\} \mid u, v \in V, u \neq v\}$.

The edges $e_1$ and $e_2$ are called multiple or parallel edges if $f(e_1) = f(e_2)$.

Note:

Edges in multigraphs are not necessarily defined as pairs, but can be of any type.

No loops are allowed in multigraphs ($u \neq v$).

# INTRODUCTION TO GRAPHS

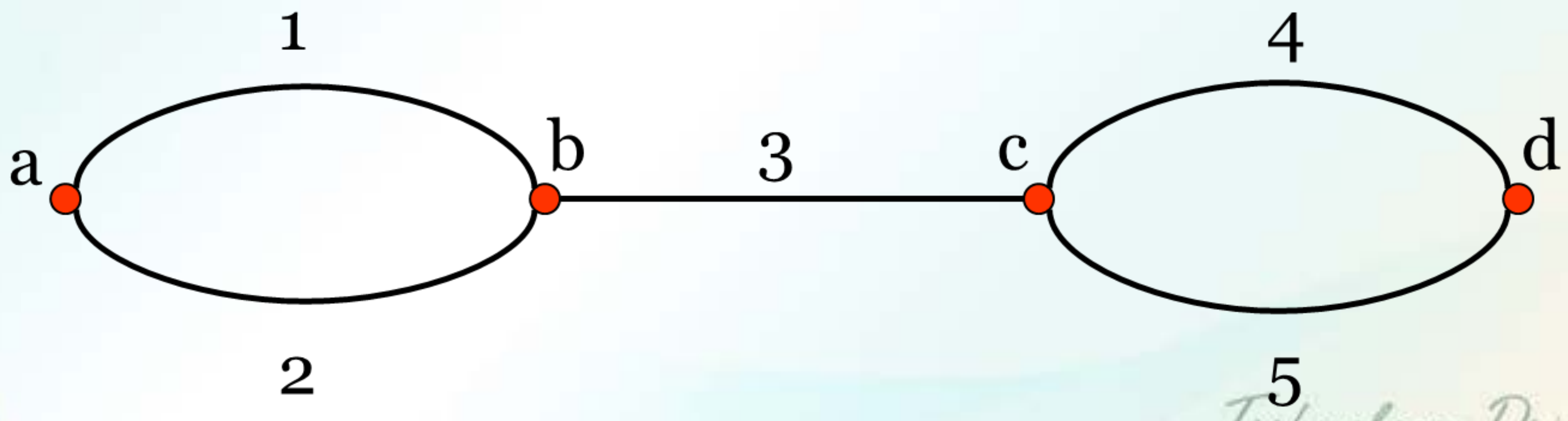
Example: A multigraph G with vertices

V = {a, b, c, d}, edges {1, 2, 3, 4, 5} and function f with

f(1) = {a, b}    f(2) = {a, b},

f(3) = {b, c},    f(4) = {c, d} and

f(5) = {c, d}:



# INTRODUCTION TO GRAPHS

To define loops, it need the following type of graph:

Definition: A pseudograph $G = (V, E)$ consists of a set $V$ of vertices, a set $E$ of edges, and a function $f$ from $E$ to $\{\{u, v\} \mid u, v \in V\}$.

An edge $e$ is a loop if $f(e) = \{u\}$ (which is a better way to denote $\{u, u\}$) for some $u \in V$.

# INTRODUCTION TO GRAPHS

**Definition: A directed graph $G = (V, E)$ consists of a set $V$ of vertices and a set $E$ of edges that are ordered pairs of elements in $V$. … leading to a new type of graph:**

**Definition: A directed multigraph $G = (V, E)$ consists of a set $V$ of vertices, a set $E$ of edges, and a function $f$ from $E$ to $\{(u, v) \mid u, v \in V\}$.**

**The edges $e_1$ and $e_2$ are called multiple edges if $f(e_1) = f(e_2)$.**

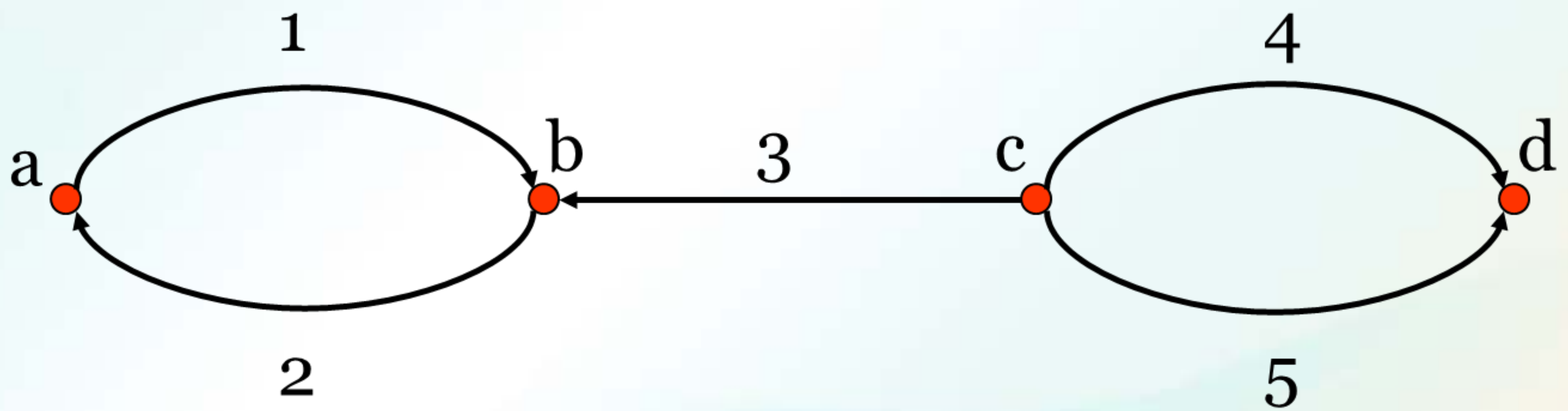
# INTRODUCTION TO GRAPHS

**Example: A directed multigraph G with vertices**

**V = {a, b, c, d}, edges {1, 2, 3, 4, 5} and function f with**

**f(1) = (a, b), f(2) = (b, a), f(3) = (c, b),**

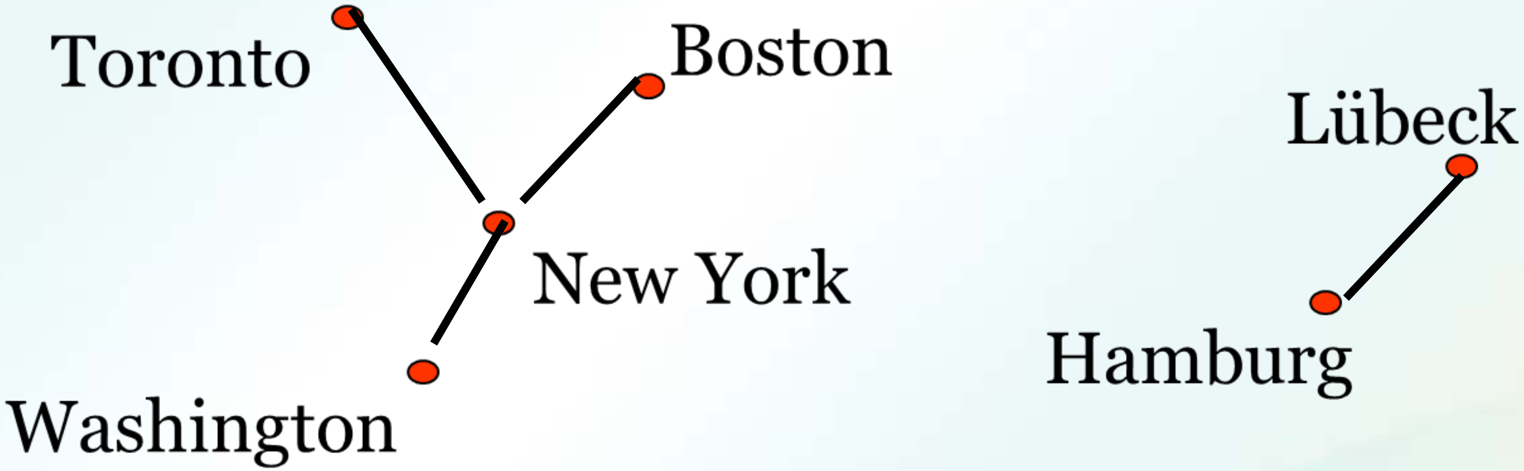
**f(4) = (c, d) and f(5) = (c, d):**



# GRAPH MODELS

**Example I: How it will represent a network of (bi-directional) railways connecting a set of cities?**

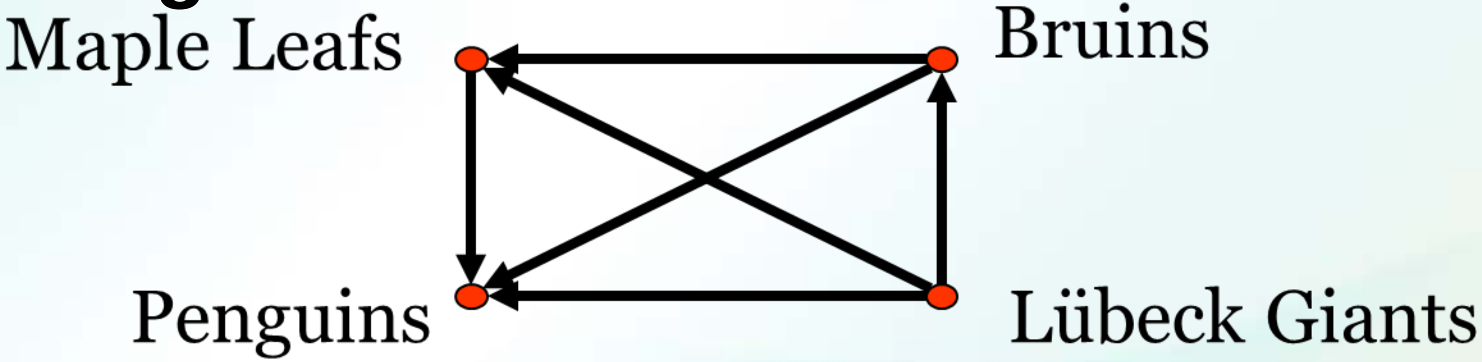
**It should use a simple graph with an edge {a, b} indicating a direct train connection between cities a and b.**



# GRAPH MODELS

**Example II: In a round-robin tournament, each team plays against each other team exactly once. How can we represent the results of the tournament (which team beats which other team)?**

**It should use a directed graph with an edge (a, b) indicating that team a beats team b.**



# GRAPH TERMINOLOGY

Definition: Two vertices u and v in an undirected graph G are called adjacent (or neighbors) in G if {u, v} is an edge in G.

If e = {u, v}, the edge e is called incident with the vertices u and v. The edge e is also said to connect u and v.

The vertices u and v are called endpoints of the edge {u, v}.

# GRAPH TERMINOLOGY

Definition: The degree of a vertex in an undirected graph is the number of edges incident with it, except that a loop at a vertex contributes twice to the degree of that vertex.

In other words, you can determine the degree of a vertex in a displayed graph by counting the lines that touch it.

The degree of the vertex v is denoted by deg(v).

# GRAPH TERMINOLOGY

`

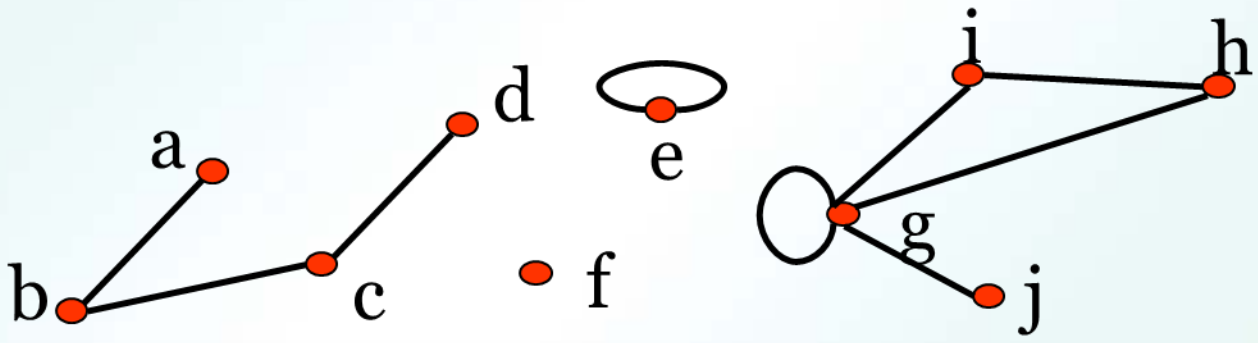
A vertex of degree 0 is called isolated, since it is not adjacent to any vertex.

Note: A vertex with a loop at it has at least degree 2 and, by definition, is not isolated, even if it is not adjacent to any other vertex.

A vertex of degree 1 is called pendant. It is adjacent to exactly one other vertex.

# GRAPH TERMINOLOGY

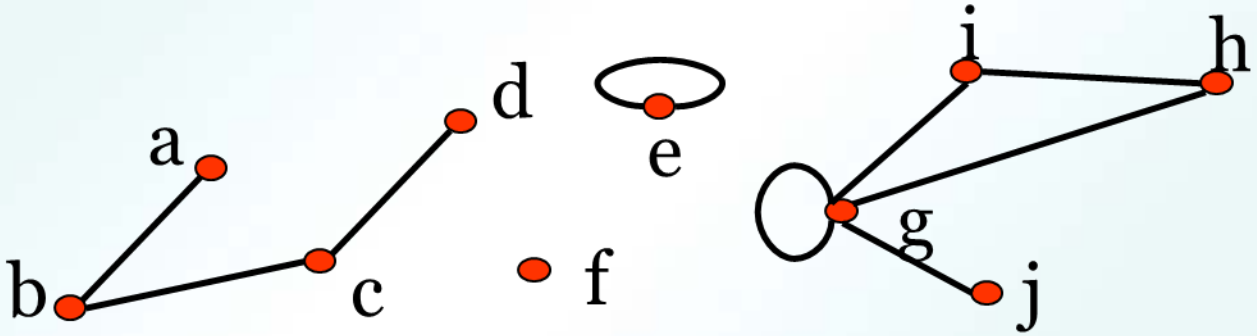
**Example: Which vertices in the following graph are isolated, which are pendant, and what is the maximum degree? What type of graph is it?**




**Solution:** Vertex f is isolated, and vertices a, d and j are pendant. The maximum degree is deg(g) = 5.
This graph is a pseudograph (undirected, loops).

# GRAPH TERMINOLOGY

**The same graph again and determine the number of its edges and the sum of the degrees of all its vertices:**




**Result:** There are 9 edges, and the sum of all degrees is 18. This is easy to explain: Each new edge increases the sum of degrees by exactly two

# GRAPH TERMINOLOGY

So if there is an even number of vertices of odd degree in the graph, it will still be even after adding an edge.

Therefore, since an undirected graph with no edges has an even number of vertices with odd degree (zero), the same must be true for any undirected graph.

# GRAPH TERMINOLOGY

**Definition: When (u, v) is an edge of the graph G with directed edges, u is said to be adjacent to v, and v is said to be adjacent from u.**

**The vertex u is called the initial vertex of (u, v), and v is called the terminal vertex of (u, v).**

**The initial vertex and terminal vertex of a loop are the same.**

# GRAPH TERMINOLOGY

**Definition: In a graph with directed edges, the in-degree of a vertex v, denoted by $deg^{-}(v)$, is the number of edges with v as their terminal vertex.**

**The out-degree of v, denoted by $deg^{+}(v)$, is the number of edges with v as their initial vertex.**

**Question: How does adding a loop to a vertex change the in-degree and out-degree of that vertex?**

**Answer: It increases both the in-degree and the out-degree by one.**

# GRAPH TERMINOLOGY

**Definition: In a graph with directed edges, the in-degree of a vertex v, denoted by $deg^-(v)$, is the number of edges with v as their terminal vertex.**

**The out-degree of v, denoted by $deg^+(v)$, is the number of edges with v as their initial vertex.**

**Question: How does adding a loop to a vertex change the in-degree and out-degree of that vertex?**

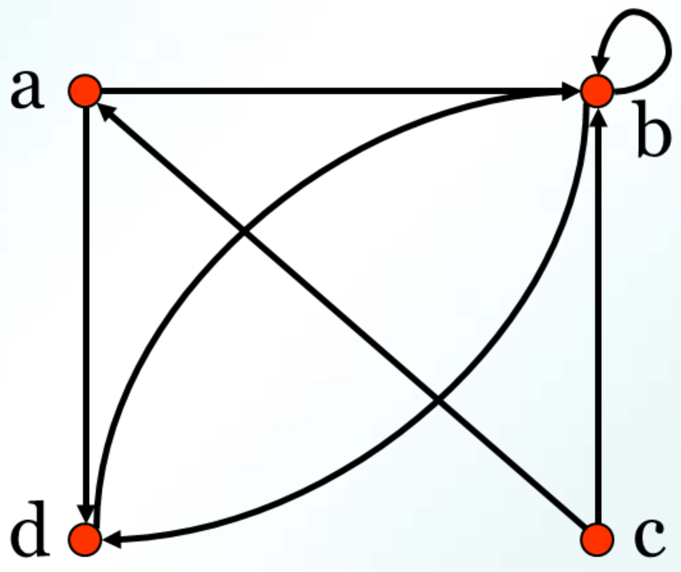
**Answer: It increases both the in-degree and the out-degree by one.**

# GRAPH TERMINOLOGY

**Example: What are the in-degrees and out-degrees of the vertices a, b, c, d in this graph:**

$\deg^-(a) = 1$

$\deg^+(a) = 2$




$\deg^-(b) = 4$

$\deg^+(b) = 2$

$\deg^-(d) = 2$

$\deg^+(d) = 1$

$\deg^-(c) = 0$

$\deg^+(c) = 2$

# GRAPH TERMINOLOGY

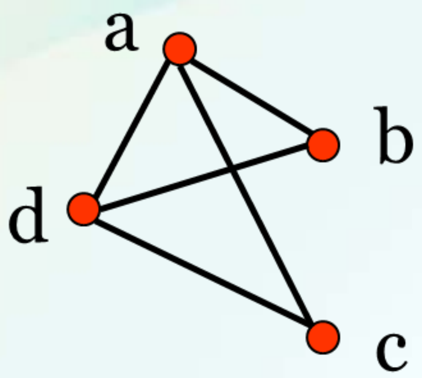
**Theorem: Let G = (V, E) be a graph with directed edges. Then:**

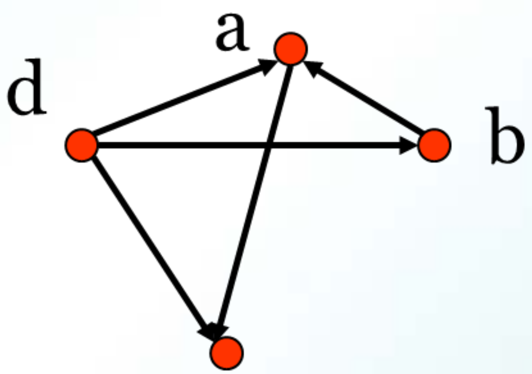
$$\sum_{v \in V} \deg^-(v) = \sum_{v \in V} \deg^+(v) = |E|$$

**This is easy to see, because every new edge increases both the sum of in-degrees and the sum of out-degrees by one.**

# REPRESENTING GRAPHS

**Graph:**




| Vertex | Adjacent Vertices |
|---|---|
| a | b, c, d |
| b | a, d |
| c | a, d |
| d | a, b, c |

| Initial Vertex | Terminal Vertices |
|---|---|
| a | c |
| b | a |
| c | |
| d | a, b, c |

# REPRESENTING GRAPHS

**Definition:** Let $G = (V, E)$ be a simple graph with $|V| = n$. Suppose that the vertices of G are listed in arbitrary order as $v_1, v_2, \ldots, v_n$.

The adjacency matrix A (or $A_G$) of G, with respect to this listing of the vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when $v_i$ and $v_j$ are adjacent, and 0 otherwise.

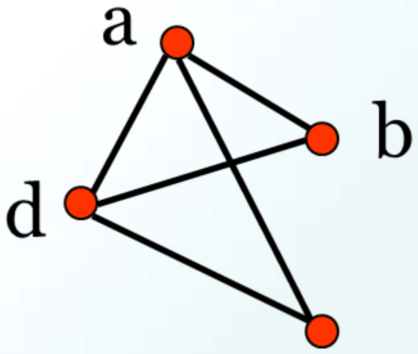
In other words, for an adjacency matrix $A = [a_{ij}]$,

$a_{ij} = 1$ if $\{v_i, v_j\}$ is an edge of G,

$a_{ij} = 0$ otherwise.

# REPRESENTING GRAPHS

**Example: What is the adjacency matrix $A_G$ for the following graph G based on the order of vertices a, b, c, d ?**




**Solution:** $$A_G = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

**Note:** Adjacency matrices of undirected graphs are always symmetric.

# REPRESENTING GRAPHS

For the representation of graphs with multiple edges, we can no longer use zero-one matrices.

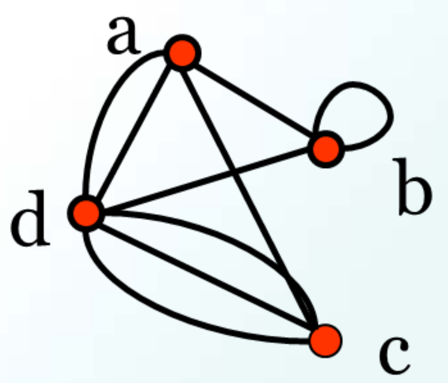
Instead, we use matrices of natural numbers.

The (i, j)th entry of such a matrix equals the number of edges that are associated with $\{v_i, v_j\}$.

# REPRESENTING GRAPHS

**Example: What is the adjacency matrix $A_G$ for the following graph G based on the order of vertices a, b, c, d ?**

**Solution:** $A_G = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 3 \\ 2 & 1 & 3 & 0 \end{bmatrix}$




**Note:** For undirected graphs, adjacency matrices are symmetric.

# REPRESENTING GRAPHS

**Definition:** Let $G = (V, E)$ be a directed graph with $|V| = n$. Suppose that the vertices of G are listed in arbitrary order as $v_1, v_2, \ldots, v_n$.

The adjacency matrix A (or $A_G$) of G, with respect to this listing of the vertices, is the $n \times n$ zero-one matrix with 1 as its (i, j)th entry when there is an edge from $v_i$ to $v_j$, and 0 otherwise.

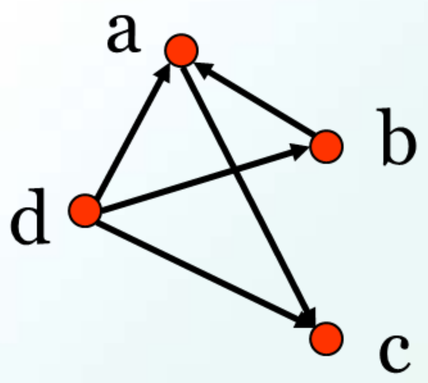
In other words, for an adjacency matrix $A = [a_{ij}]$,

$a_{ij} = 1$ if $(v_i, v_j)$ is an edge of G,
$a_{ij} = 0$ otherwise.

# REPRESENTING GRAPHS

**Example: What is the adjacency matrix $A_G$ for the following graph G based on the order of vertices a, b, c, d ?**

**Solution:** $$A_G = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$



# REPRESENTING GRAPHS

Definition: Let $G = (V, E)$ be an undirected graph with $|V| = n$ and $|E| = m$. Suppose that the vertices and edges of $G$ are listed in arbitrary order as $v_1, v_2, \ldots, v_n$ and $e_1, e_2, \ldots, e_m$, respectively.

The incidence matrix of G with respect to this listing of the vertices and edges is the $n \times m$ zero-one matrix with 1 as its $(i, j)$th entry when edge $e_j$ is incident with $v_i$, and 0 otherwise.

In other words, for an incidence matrix $M = [m_{ij}]$,

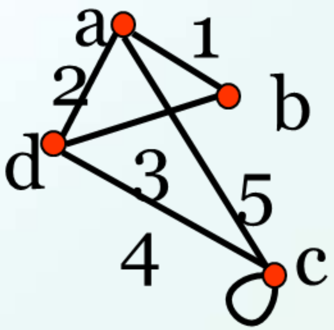
$m_{ij} = 1$ if edge $e_j$ is incident with $v_i$
$m_{ij} = 0$ otherwise.

# REPRESENTING GRAPHS

**Example: What is the incidence matrix M for the following graph G based on the order of vertices a, b, c, d and edges 1, 2, 3, 4, 5, 6?**

**Solution:**

$$M = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$




**Note:** Incidence matrices of directed graphs contain two 1s per column for edges connecting two vertices and one 1 per column for loops.

# TREES

Definition: A tree is a connected undirected graph with no simple circuits.

Since a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops.

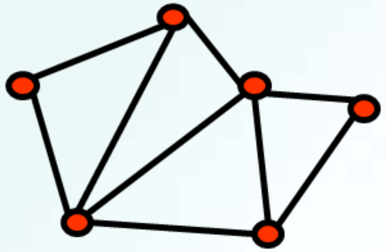
Therefore, any tree must be a simple graph.

Theorem: An undirected graph is a tree if and only if there is a unique simple path between any of its vertices.

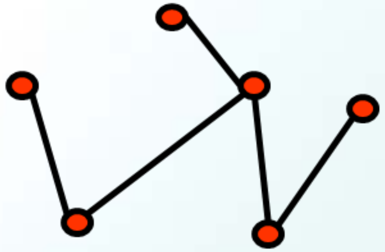
# TREES

## Trees:

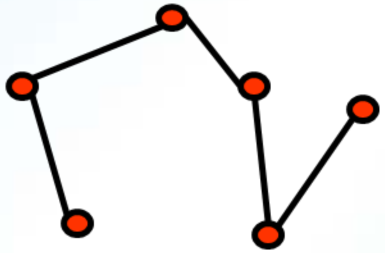
- **Example:** Are the following graphs trees?



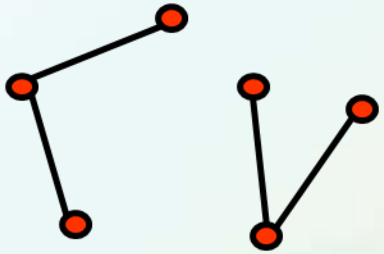
No.



Yes.



Yes.



No.

# TREES

**Definition: An undirected graph that does not contain simple circuits and is not necessarily connected is called a forest.**

**In general, we use trees to represent hierarchical structures. We often designate a particular vertex of a tree as the root. Since there is a unique path from the root to each vertex of the graph, we direct each edge away from the root.**

**Thus, a tree together with its root produces a directed graph called a rooted tree.**

# TREE TERMINOLOGY

If v is a vertex in a rooted tree other than the root, the parent of v is the unique vertex u such that there is a directed edge from u to v.

When u is the parent of v, v is called the child of u.

Vertices with the same parent are called siblings.

The ancestors of a vertex other than the root are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root.

# TREE TERMINOLOGY

**The descendants of a vertex v are those vertices that have v as an ancestor.**

**A vertex of a tree is called a leaf if it has no children.**

**Vertices that have children are called internal vertices.**

**If a is a vertex in a tree, then the subtree with a as its root is the subgraph of the tree consisting of a and its descendants and all edges incident to these descendants.**

# TREE TERMINOLOGY

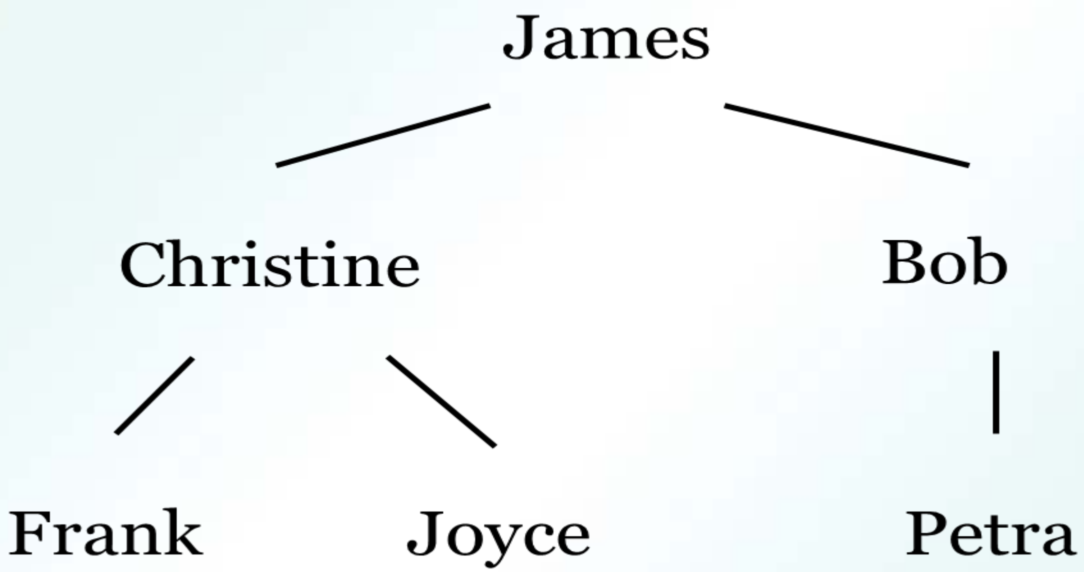
The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex.

The level of the root is defined to be zero.

The height of a rooted tree is the maximum of the levels of vertices.

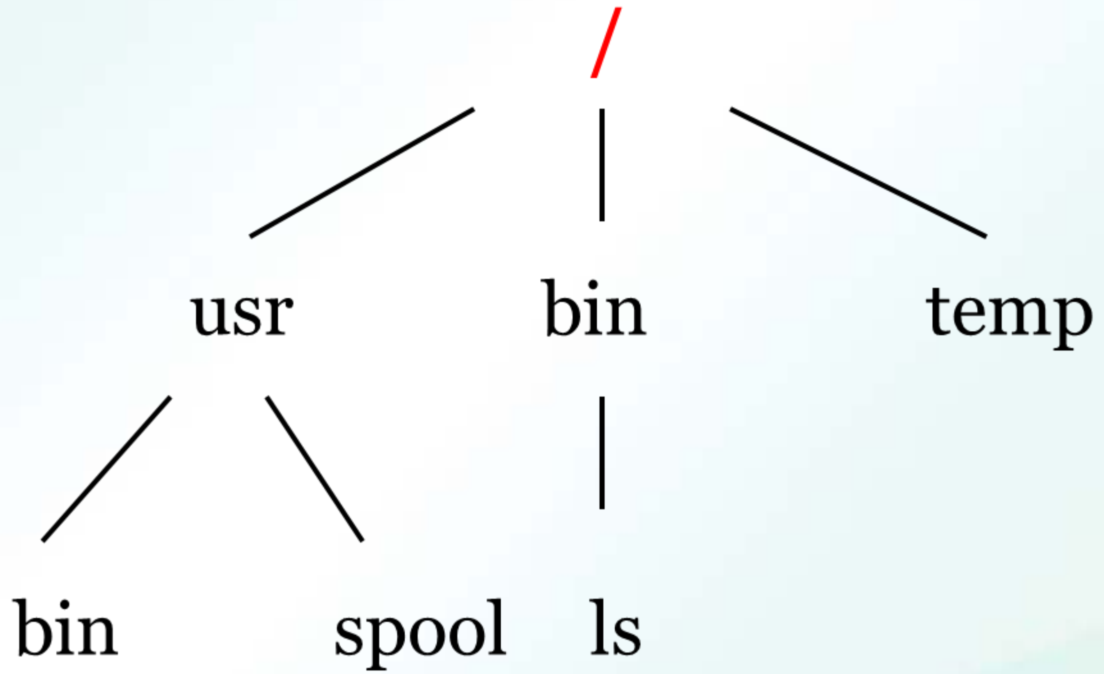
# TREES

## Example I: Family tree



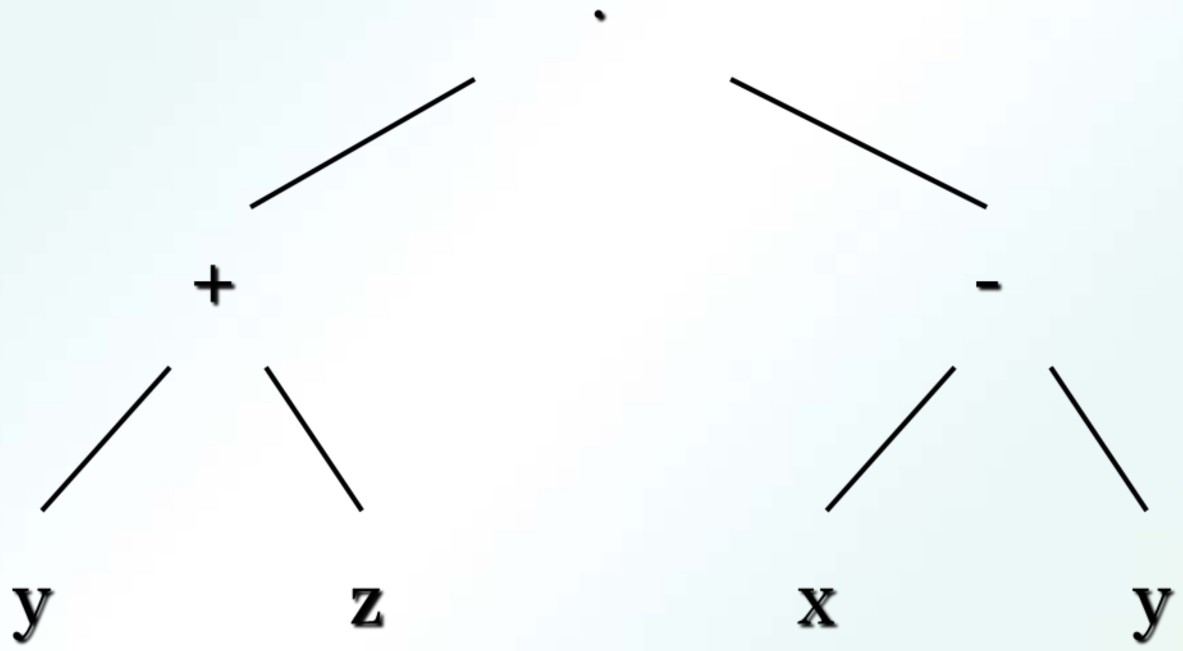
# TREES

## Example II: File System



# TREES

## Example III: Arithmetic Expressions




This tree represents the expression $(y + z)\cdot(x - y)$.

# TREES

**Definition: A rooted tree is called an m-ary tree if every internal vertex has no more than m children.**

**The tree is called a full m-ary tree if every internal vertex has exactly m children.**

**An m-ary tree with $m = 2$ is called a binary tree.**

**Theorem: A tree with n vertices has $(n – 1)$ edges.**

**Theorem: A full m-ary tree with i internal vertices contains $n = mi + 1$ vertices.**

# BINARY SEARCH TREES

If want to perform a large number of searches in a particular list of items, it can be worthwhile to arrange these items in a binary search tree to facilitate the subsequent searches.

A binary search tree is a binary tree in which each child of a vertex is designated as a right or left child, and each vertex is labeled with a key, which is one of the items.

When we construct the tree, vertices are assigned keys so that the key of a vertex is both larger than the keys of all vertices in its left subtree and smaller than the keys of all vertices in its right subtree.

# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**

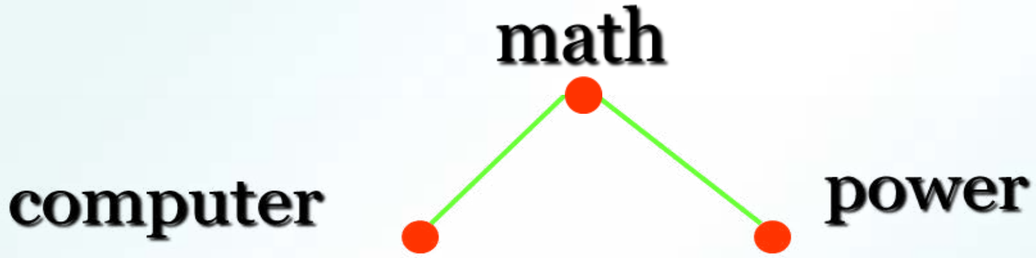
- **math**

# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**

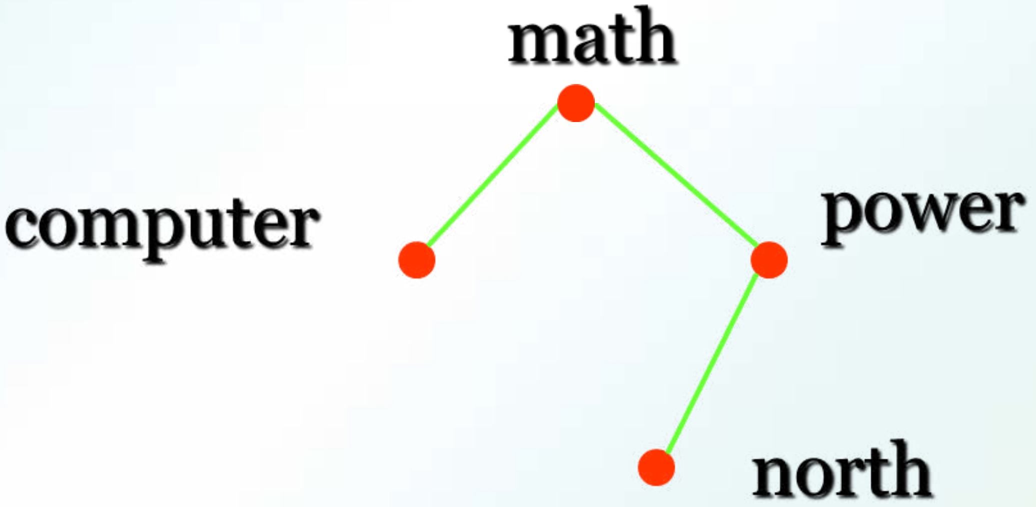
# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**



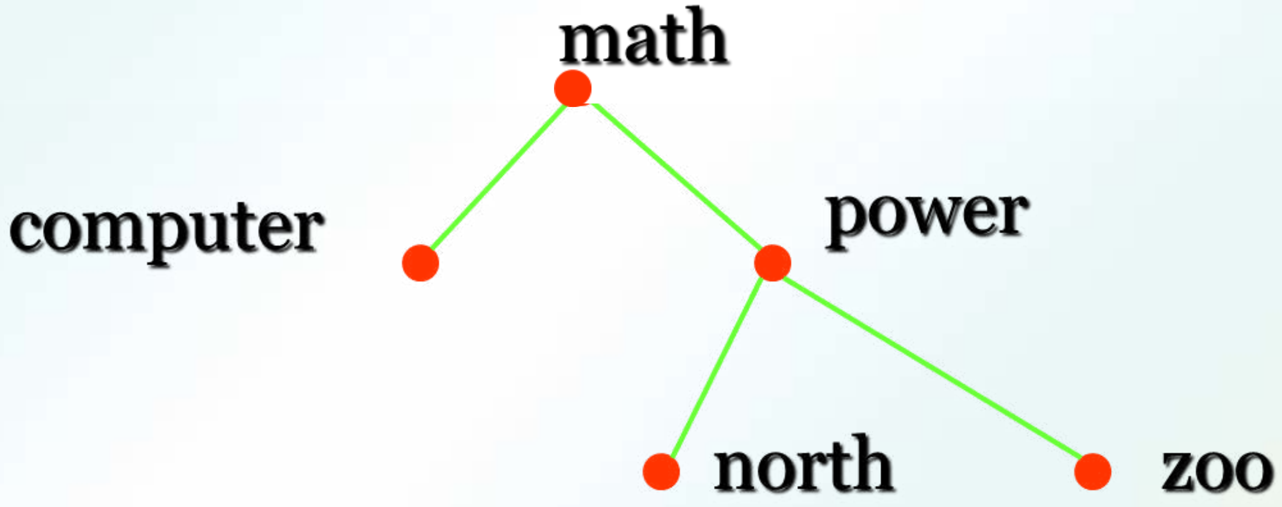
# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**



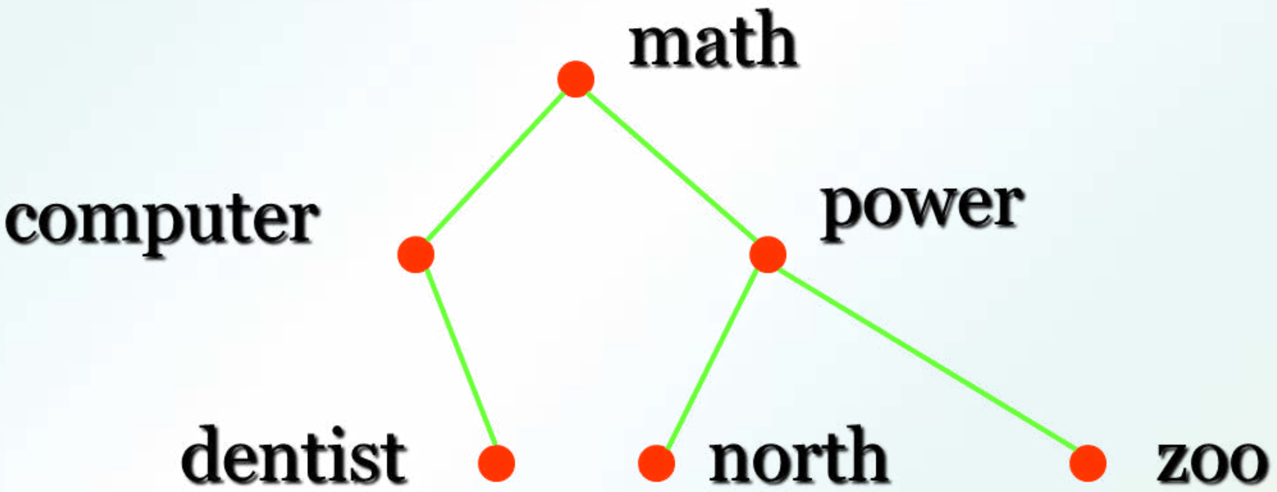
# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**



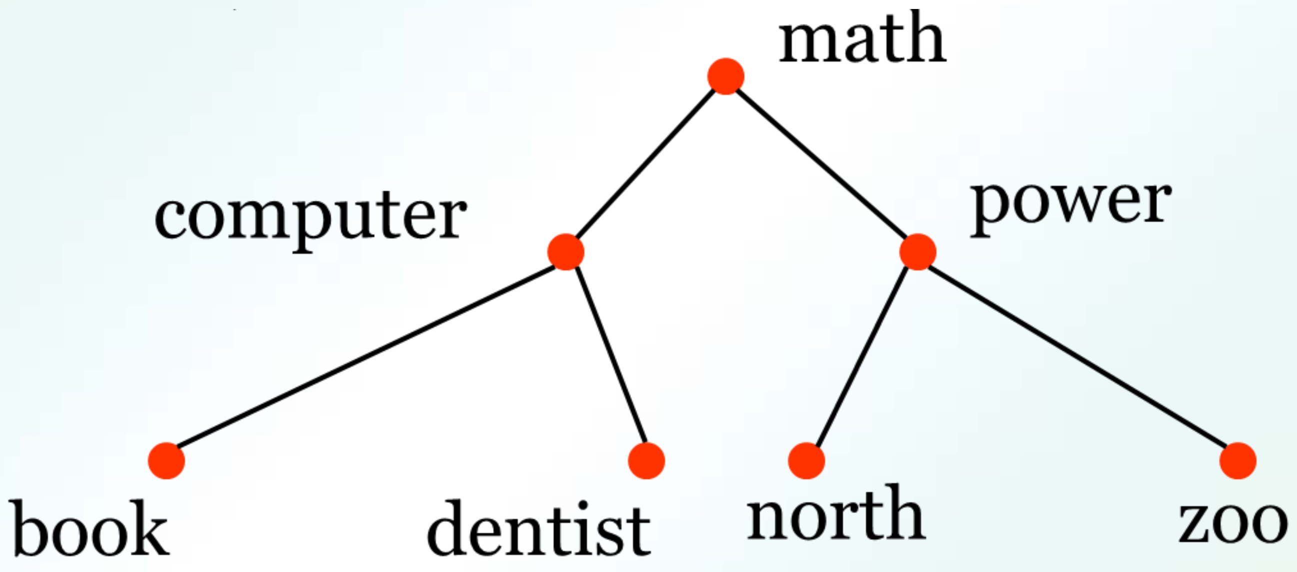
# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**



# BINARY SEARCH TREES

**Example: Construct a binary search tree for the strings math, computer, power, north, zoo, dentist, book.**



# BINARY SEARCH TREES

To perform a search in such a tree for an item x, it can start at the root and compare its key to x. If x is less than the key, we proceed to the left child of the current vertex, and if x is greater than the key, we proceed to the right one.

This procedure is repeated until we either found the item were looking for, or it cannot proceed any further.

In a balanced tree representing a list of n items, search can be performed with a maximum of $\lceil \log(n + 1) \rceil$ steps (compare with binary search).

# BINARY SEARCH TREES

**Example:**
**Build a binary search tree for the words *oenology*, *phrenology*, *campanology*, *ornithology*, *ichthyology*, *limnology*, *alchemy*, and *astrology* using alphabetical order**

# SEATWORK:

- **1/4 yellow pad**
- **Build a binary search tree for the words *banana*, *peach*, *apple*, *pear*, *coconut*, *mango*, and *papaya* using alphabetical order.**

# SPANNING TREES

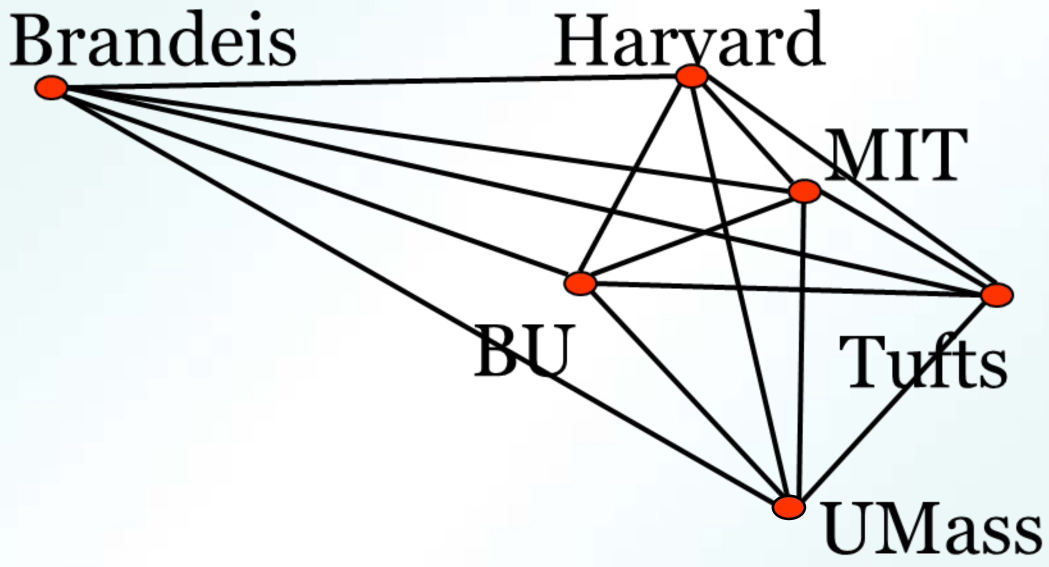
**Definition: Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G.**

**Note: A spanning tree of G = (V, E) is a connected graph on V with a minimum number of edges (|V| - 1).**

**Example: Since winters in Boston can be very cold, six universities in the Boston area decide to build a tunnel system that connects their libraries.**

# SPANNING TREES

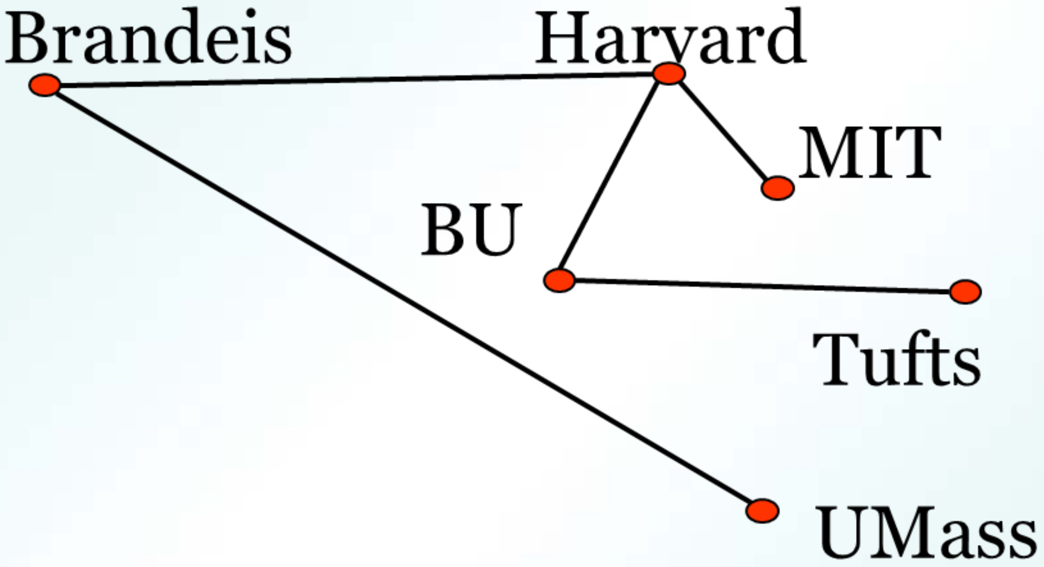
## The complete graph including all possible tunnels:




The spanning trees of this graph connect all libraries with a minimum number of tunnels

# SPANNING TREES

**Example for a spanning tree:**




Since there are 6 libraries, 5 tunnels are sufficient to connect all of them

# SPANNING TREES

Now imagine that you are in charge of the tunnel project. How can you determine a tunnel system of minimal cost that connects all libraries?

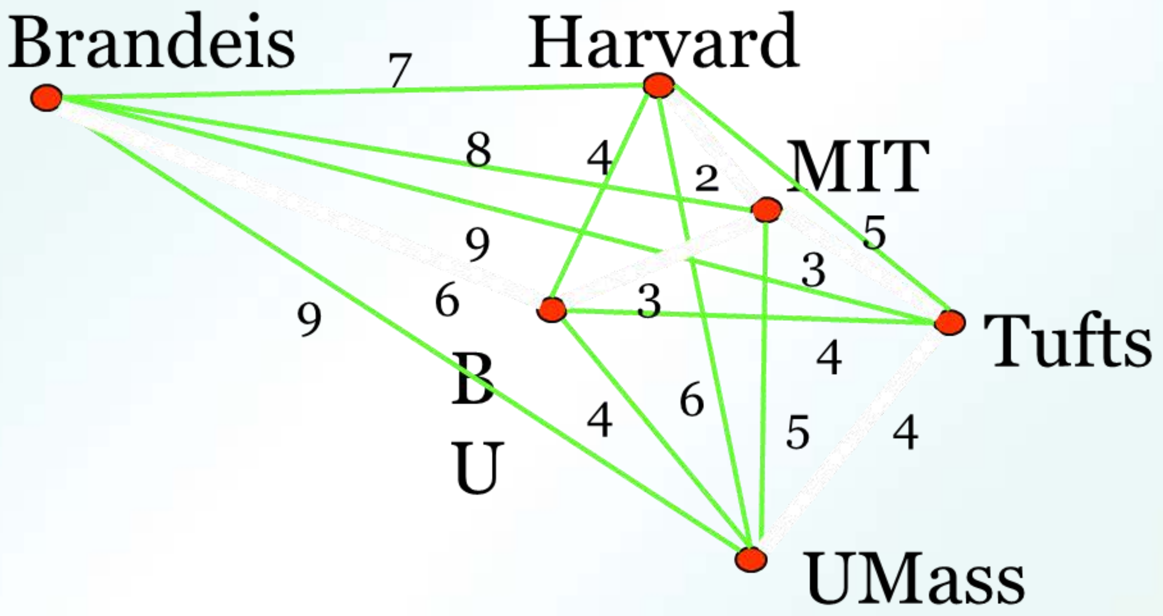
Definition: A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges.

How can we find a minimum spanning tree?

# SPANNING TREES

**Spanning:**

- The complete graph with cost labels (in billion $):




The least expensive tunnel system costs $20 billion.

# SPANNING TREES

**Prim's Algorithm:**

Begin by choosing any edge with smallest weight and putting it into the spanning tree,

successively add to the tree edges of minimum weight that are incident to a vertex already in the tree and not forming a simple circuit with those edges already in the tree,

stop when (n – 1) edges have been added.

# SPANNING TREES

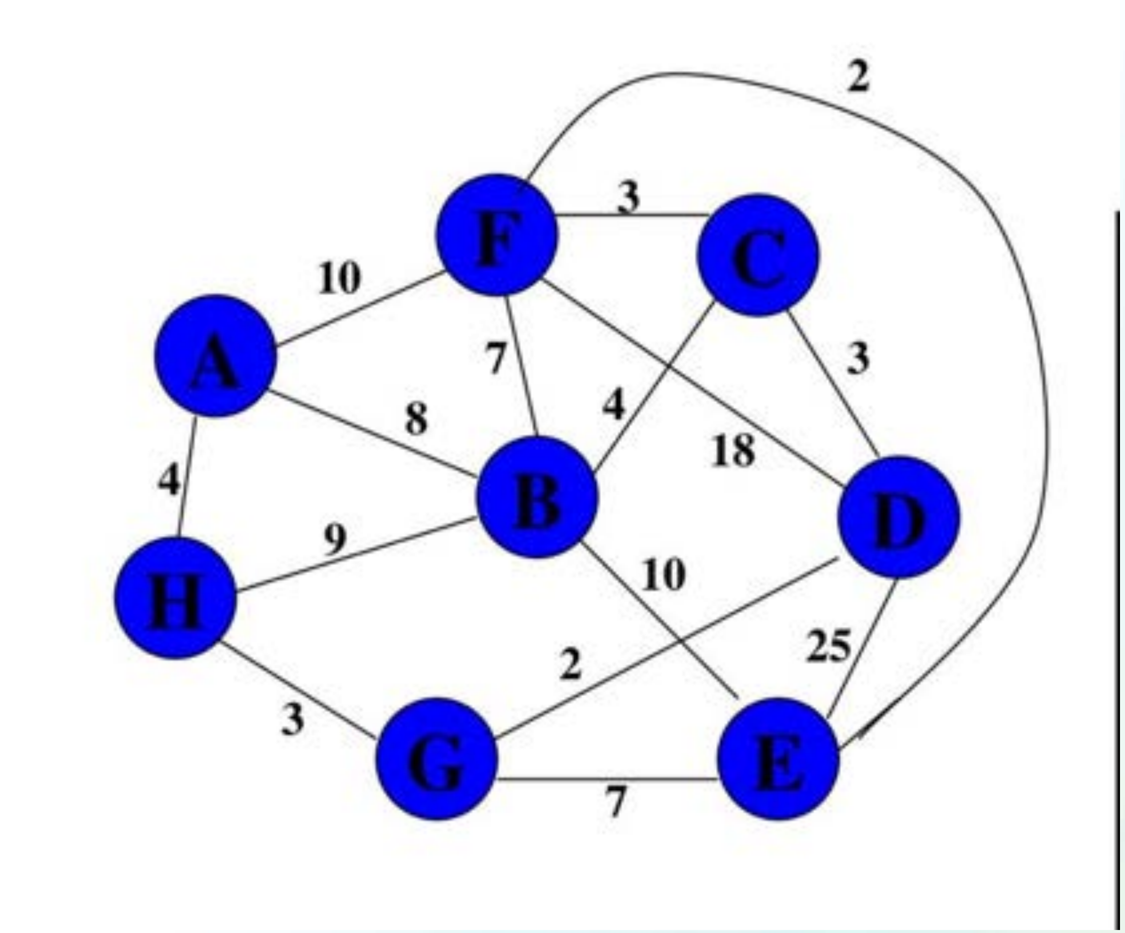
**Kruskal's Algorithm:**

**Kruskal's algorithm is identical to Prim's algorithm, except that it does not demand new edges to be incident to a vertex already in the tree.**

**Both algorithms are guaranteed to produce a minimum spanning tree of a connected weighted graph.**

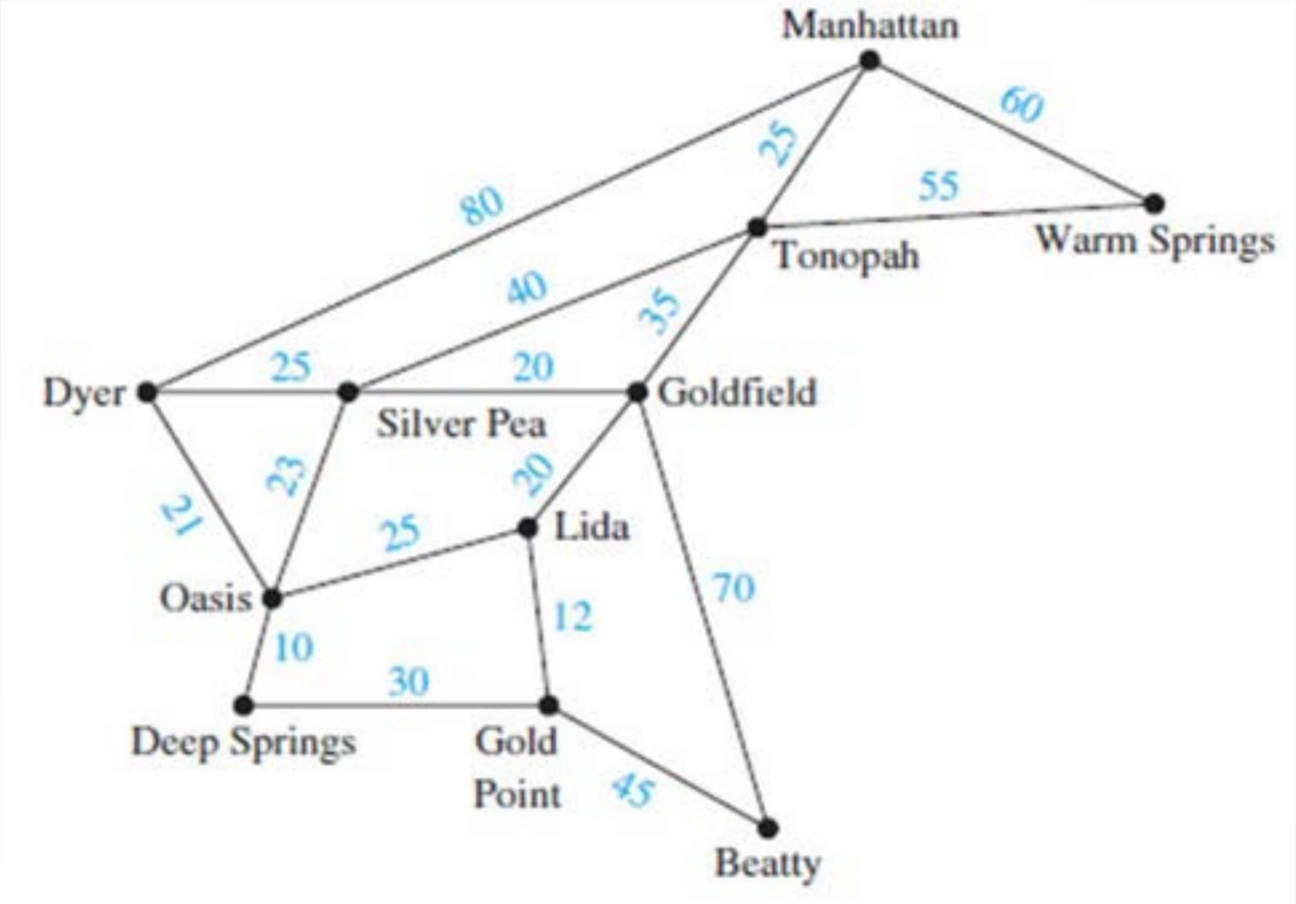
# SPANNING TREES

**Spanning**



# SPANNING TREES

## SEATWORK:



# References


https://www.sanfoundry.com/data-structure

http://www.bowdoin.edu/~ltoma/teaching/cs210/spring09/Slides/210-Trees.pdf

https://www.youtube.com/watch?v=uvxmgr5Doao

https://www.scribd.com/presentation/357823185/Binary-Search-Tree

https://www.tutorialspoint.com/discrete_mathematics/introduction_to_trees.htm

https://www.slideshare.net/ifadislam7/discrete-math

https://www.javatpoint.com/discrete-mathematics-minimum-spanning-tree

http://discrete.openmathbooks.org/dmoi3/sec_trees.html