BACKGROUND: A BRIEF INTRODUCTION TO GRAPH THEORY

- **General definitions; Representations;**
- **Graph Traversals;**
- **Topological sort;**

Graphs – definitions & representations

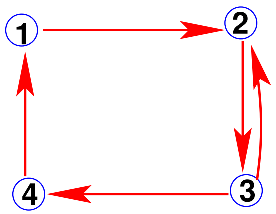
➤ Graph theory is a fundamental tool in sparse matrix techniques.

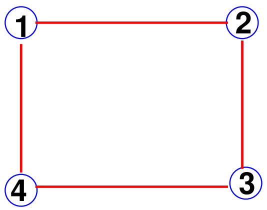
DEFINITION. A graph G is defined as a pair of sets $G = (V, E)$ with $E \subset V \times V$. So G represents a binary relation. The graph is undirected if the binary relation is symmetric. It is directed otherwise. V is the vertex set and E is the edge set.

If R is a binary relation between elements in V then, we can represent it by a graph $G = (V, E)$ as follows:

$$(u, v) \in E \leftrightarrow u \; R \; v$$

Undirected graph $\leftrightarrow$ symmetric relation





(1 R 2); (4 R 1); (2 R 3); (3 R 2); (3 R 4) | (1 R 2); (2 R 3); (3 R 4); (4 R 1)

Given the numbers 5, 3, 9, 15, 16, show the two graphs representing the relations

R1: Either $x < y$ or y divides x.

R2: x and y are congruent modulo 3. [mod(x,3) = mod(y,3)]

➤ $|E| \leq |V|^2$. For undirected graphs: $|E| \leq |V|(|V| + 1)/2$.

➤ A sparse graph is one for which $|E| \ll |V|^2$.

Graphs – Examples and applications

➤ Applications of graphs are numerous.

1. Airport connection system: (a) R (b) if there is a non-stop flight from (a) to (b).
2. Highway system;
3. Computer Networks;
4. Electrical circuits;
5. Traffic Flow;
6. Sparse matrix computations;

...

Basic Terminology & notation:

➤ If $(u, v) \in E$, then v is adjacent to u. The edge (u, v) is incident to u and v.

➤ If the graph is directed, then (u, v) is an outgoing edge from u and incoming edge to v

➤ $Adj(i) = \{j | j \text{ adjacent to } i\}$

➤ The degree of a vertex v is the number of edges incident to v. Can also define the indegree and outdegree. (Sometimes self-edge $i \to i$ omitted)

➤ $|S|$ is the cardinality of set S [so $|Adj(i)| ==$ deg(i)]

➤ A subgraph $G' = (V', E')$ of G is a graph with $V' \subset V$ and $E' \subset E$.

Representations of Graphs

➤ A graph is nothing but a collection of vertices (indices from 1 to n), each with a set of its adjacent vertices [in effect a 'sparse matrix without values']

➤ Therefore, can use any of the sparse matrix storage formats - omit the real values arrays.

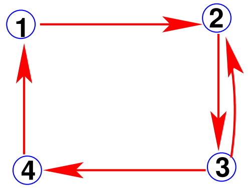
Adjacency matrix Assume $V = \{1, 2, \cdots, n\}$. Then the adjacency matrix of $G = (V, E)$ is the $n \times n$ matrix, with entries:

$$a_{i,j} = \begin{cases} 1 \text{ if } (i, j) \in E \\ 0 \text{ Otherwise} \end{cases}$$

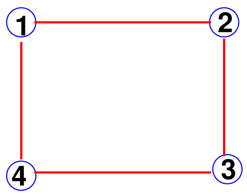
Representations of Graphs (cont.)

Example:

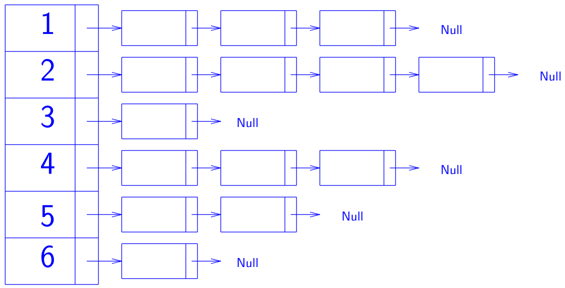
$$\begin{bmatrix} & 1 & & \\ & & 1 & \\ & 1 & & 1 \\ 1 & & & \end{bmatrix}$$



$$\begin{bmatrix} & 1 & & 1 \\ 1 & & 1 & \\ & 1 & & 1 \\ 1 & & 1 & \end{bmatrix}$$



Dynamic representation: Linked lists



➤ An array of linked lists. A linked list associated with vertex i, contains all the vertices adjacent to vertex i.

➤ General and concise for 'sparse graphs' (the most practical situations).

➤ Not too economical for use in sparse matrix methods

More terminology & notation

➤ For a given $Y \subset X$, the section graph of Y is the subgraph $G_Y = (Y, E(Y))$ where

$$E(Y) = \{(x, y) \in E | x \in Y, \quad y \ in \ Y\}$$

➤ A section graph is a clique if all the nodes in the subgraph are pairwise adjacent ($\rightarrow$ dense block in matrix)

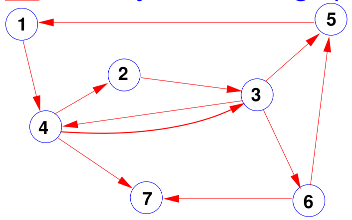
➤ A path is a sequence of vertices $w_0, w_1, \dots, w_k$ such that $(w_i, w_{i+1}) \in E$ for $i = 0, \dots, k-1$.

➤ The length of the path $w_0, w_1, \dots, w_k$ is k (# of edges in the path)

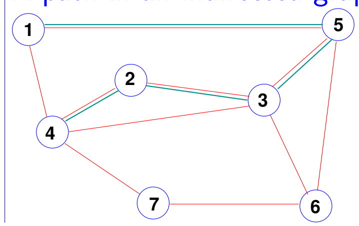
➤ A cycle is a closed path, i.e., a path with $w_k = w_0$.

➤ A graph is acyclic if it has no cycles.

Find cycles in this graph:



A path in an indirected graph



➤ A path $w_0, \dots, w_k$ is simple if the vertices $w_0, \dots, w_k$ are distinct (except that we may have $w_0 = w_k$ for cycles).

➤ An undirected graph is connected if there is path from every vertex to every other vertex.

➤ A digraph with the same property is said to be strongly connected

➤ The undirected form of a directed graph the undirected graph obtained by removing the directions of all the edges.

➤ Another term used "symmetrized" form -

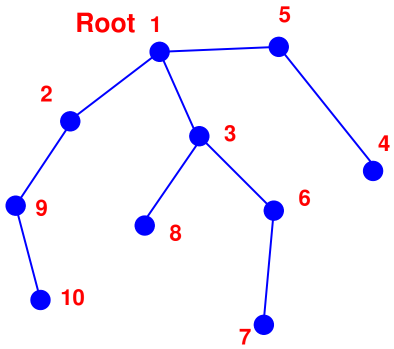
➤ A <u>directed</u> graph whose undirected form is connected is said to be weakly connected or connected.

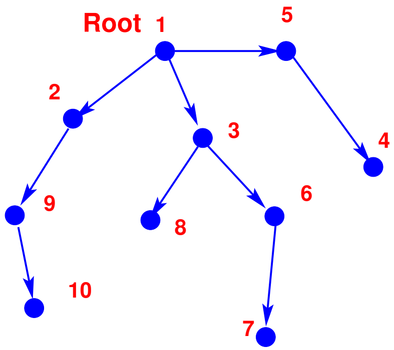
➤ Tree = a graph whose undirected form, i.e., symmetrized form, is acyclic & connected

➤ Forest = a collection of trees

➤ In a rooted tree one specific vertex is designated as a root.

➤ Root determines orientation of the tree edges in parent-child relation





➤ Parent-Child relation: immediate neighbors of root are children. Root is their parent. Recursively define children-parents

➤ In example: v_3 is parent of v_6, v_8 and v_6, v_8 are chidren of v_3.

➤ Nodes that have no children are leaves. In example: v_{10}, v_7, v_8, v_4

➤ Descendent, ancestors, ...

Tree traversals

➤ Tree traversal is a process of visiting all vertices in a tree. Typically traversal starts at root.

➤ Want: systematic traversals of all nodes of tree – moving from a node to a child or parent

➤ Preorder traveral: Visit children before parent [recursively]

In example: $v_1, v_2, v_9, v_{10}, v_3, v_8, v_6, v_7, v_5, v_4$

➤ Preorder traveral: Visit parent then children [recursively]

In example : $v_{10}, v_9, v_2, v_8, v_7, v_6, v_3, v_4, v_5, v_1$

Graphs Traversals – Depth First Search

➤ Issue: systematic way of visiting all nodes of a general graph

➤ Two basic methods: Breadth First Search (to be seen later) and Depth-First Search

➤ Idea of DFS is recursive:

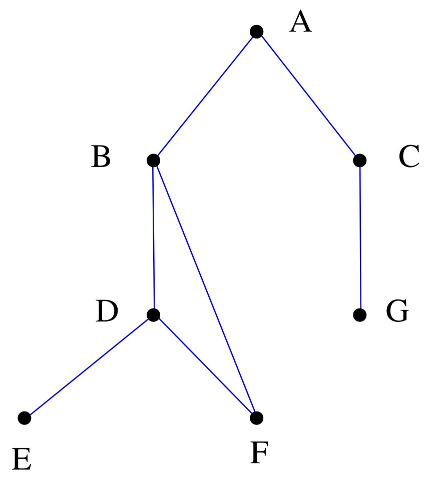
Algorithm $DFS(G, v)$ (DFS from v)

- Visit and Mark v;
- for all edges (v, w) do
  - if w is not marked then $DFS(G, w)$

➤ If G is undirected and connected, all nodes will be visited

➤ If G is directed and strongly connected, all nodes will be visited

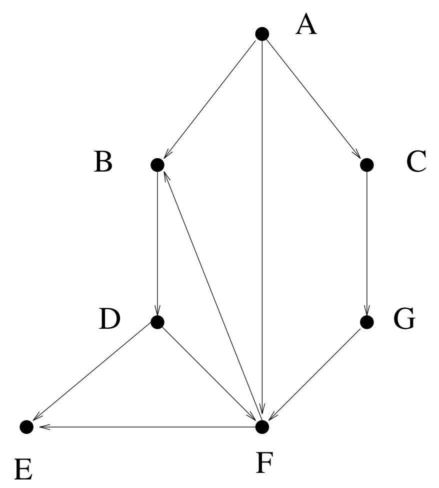
Depth First Search – undirected graph example



➤ Assume adjacent nodes are listed in alphabetical order.

DFS traversal from A: ?

Depth First Search – directed graph example



➤ Assume adjacent nodes are listed in alphabetical order.

DFS traversal from A: ?

Depth-First-Search Tree: Consider the parent-child relation: v is a parent of u if u was visited from v in the depth first search algorithm. The (directed) graph resulting from this binary relation is a tree called the Depth-First-Search Tree. To describe tree: only need the parents list.

➤ To traverse all the graph we need a DFS(v,G) from each node v that has not been visited yet – so add another loop. Refer to this as

DFS(G)

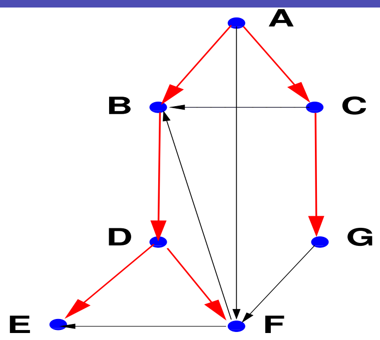
➤ When a new vertex is visited in DFS, some work is done. Example: we can build a stack of nodes visited to show order (reverse order: easier) in which the node is visited.


```
EXAMPLE         3     5       We assume adjacency list
                o     o       is in increasing order.
                |     |       [e.g: Adj(4)=(1,5,6,7)]
                |     |
                |     |4    6
              1 o-----o-----o
               /\     |    /
              /  \    |   /
             /    \   |  /
            /      \  | /
         2 o      7  \|/
                      o

DFS traversal: 1 --> 2 --> 3 --> 4  --> 5 --> 6 --> 7
Parents list:        1     1     1      4     4     6

             3  o       o 5
                |       |
                |       |4      6
              1 o-------o-------o      <-----  Depth First
               /               /               Search Tree
              /               /
           2 o           7   o
```

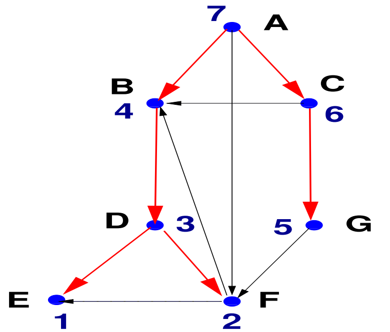
Back edges, forward edges, and cross edges



- Thick red lines: DFS traversal tree from A
- $A \rightarrow F$ is a Forward edge
- $F \rightarrow B$ is a Back edge
- $C \rightarrow B$ and $G \rightarrow F$ are Cross-edges.

Postorder traversal : label the nodes so that children in tree labeled before root.

- Important for some algorithms
- label(i) == order of completion of visit of subtree rooted at node i



- Notice:
  - Tree-edges / Forward edges : labels decrease in $\longrightarrow$
  - Cross edges : labels decrease in $\longrightarrow$
  - Back-edges : labels increase in $\longrightarrow$

Properties of Depth First Search

- If G is a connected undirected (or strongly directed) graph, then each vertex will be visited once and each edge will be inspected at least once.
- Therefore, for a connected undirected graph, The cost of DFS is $O(|V| + |E|)$
- If the graph is undirected, then there are no cross-edges. (all non-tree edges are called 'back-edges')

Theorem: A directed graph is acyclic iff a DFS search of G yields no back-edges.

Topological Sort

The Problem: Given a Directed Acyclic Graph (DAG), order the vertices from 1 to n such that, if (u, v) is an edge, then u appears before v in the ordering.

- Equivalently, label vertices from 1 to n so that in any (directed) path from a node labelled k, all vertices in the path have labels $>k$.
- Many Applications
- Prerequisite requirements in a program
- Scheduling of tasks for any project
- Parallel algorithms;
- ...

Topological Sorting: A first algorithm

Property exploited: An acyclic Digraph must have at least one vertex with indegree = 0.

✍ Prove this

Algorithm:

➤ First label these vertices as 1, 2, . . . , k;

➤ Remove these vertices and all edges incident from them

➤ Resulting graph is again acyclic ... ∃ nodes with indegree $= 0$. label these nodes as $k + 1, k + 2, \ldots,$

➤ Repeat..

✍ Explore implementation aspects.

Alternative methods: Topological sort from DFS

➤ Depth first search traversal of graph.

➤ Do a 'post-order traversal' of the DFS tree.

Algorithm $Lst = Tsort(G)$ (post-order DFS from v)

```
Mark = zeros(n,1);   Lst = ∅
for v=1:n do:
   if (Mark(v)== 0)
      [Lst, Mark] = dfs(v, G, Lst, Mark);
   end
end
```

➤ dfs(v, G, Lst, Mark) is the DFS(G,v) which adds v to the top of Lst after finishing the traversal from v

Lst = DFS(G,v)

- Visit and Mark v;
- for all edges (v, w) do
  - if w is not marked then $Lst = DFS(G, w)$
- $Lst = [v, Lst]$

➤ Topological order given by the final Lst array of Tsort

✍ Explore implementation issue

✍ Implement in matlab

✍ Show correctness [i.e.: is this indeed a topol. order? hint: no back-edges in a DAG]

GRAPH MODELS FOR SPARSE MATRICES

- **See Chap. 3 of text**
- **Sparse matrices and graphs.**
- **Bipartite model**
- **Graph Laplaceans. Application: Graph Partitioning**

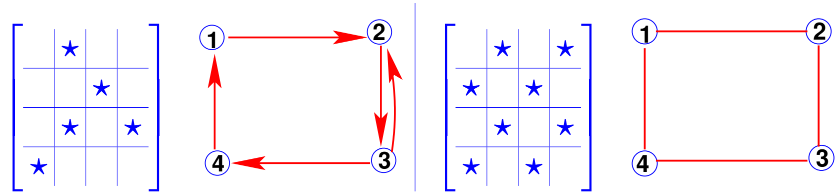
Graph Representations of Sparse Matrices. Recall:

Adjacency Graph $G = (V, E)$ of an $n \times n$ matrix A :

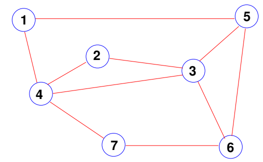
$$V = \{1, 2, ..., N\} \qquad E = \{(i, j) | a_{ij} \neq 0\}$$

- G == undirected if A has a symmetric pattern

Example:



✍ Show the matrix pattern for the graph on the right and give an interpretation of the path v_4, v_2, v_3, v_5, v_1 on the matrix



- A separator is a set Y of vertices such that the graph G_{X-Y} is disconnected.

Example: $Y = \{v_3, v_4, v_5\}$ is a separator in the above figure

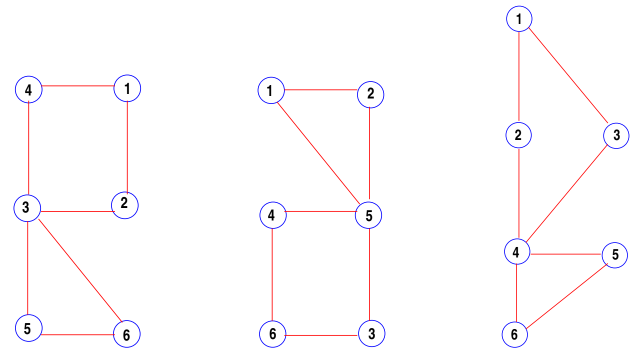
Example: Adjacency graph of:

$$A = \begin{bmatrix} & \star & & \star & & \\ \star & & \star & & & \\ & \star & & \star & \star & \star \\ \star & & \star & & & \\ & & \star & & & \star \\ & & \star & & \star & \end{bmatrix}.$$

Example: For any matrix A, what is the graph of A^2? [interpret in terms of paths in the graph of A]

- Two graphs are isomorphic is there is a mapping between the vertices of the two graphs that preserves adjacency.

✍ Are the following 3 graphs isomorphic? If yes find the mappings between them.



- Graphs are identical – labels are different

Bipartite graph representation

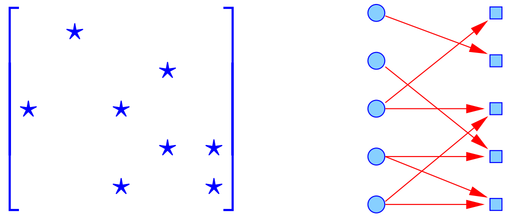
➤ Each row is represented by a vertex

➤ Each column is represented by a vertex

➤ Relations only between rows and columns.

➤ Row i is connected to column j if $a_{ij} \neq 0$

Example:



Interpretation of graphs of matrices

➤ Note: the bipartite model is used only for specific cases [e.g. rectangular matrices, ...] - By default we use the standard definition of graphs.

✍ What is the graph of $A + B$ (for two $n \times n$ matrices)?

✍ What is the graph of A^T ?

✍ What is the graph of $A.B$?

✍ What is the graph of A^k?

✍ In which of the following cases is the underlying physical mesh the same as the graph of A (in the sense that edges are the same):

- Finite difference mesh [consider the simple case of 5-pt and 7-pt FD problems - then 9-point meshes.]
- Finite element mesh with linear elements (e.g. triangles)?
- Finite element mesh with other types of elements? [to answer this question you would have to know more about higher order elements]

Graph Laplaceans - Definition

➤ "Laplace-type" matrices associated with general undirected graphs – useful in many applications

➤ Given a graph $G = (V, E)$ define

- A matrix W of weights w_{ij} for each edge
- Assume $w_{ij} \geq 0,$, $w_{ii} = 0$, and $w_{ij} = w_{ji}\ \forall(i, j)$
- The diagonal matrix $D = diag(d_i)$ with $d_i = \sum_{j\neq i} w_{ij}$

➤ Corresponding *graph Laplacean* of G is:

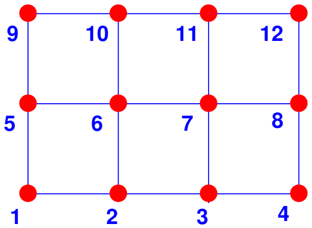
$$L = D - W$$

➤ Gershgorin's theorem $\rightarrow$ L is positive semidefinite

➤ Simplest case:

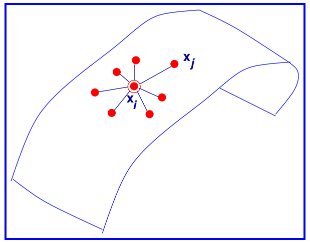
$$w_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \& i \neq j \\ 0 \quad \text{else} \end{cases} \qquad D = \text{diag}\left[d_i = \sum_{j \neq i} w_{ij} \right]$$

✍ Define the graph Laplacean for the graph associated with the simple mesh shown next. [use the simple weights of 0 or 1]



✍ What is the difference with the discretization of the Laplace operator in 2-D for case when mesh is the same as this graph?

A few properties of graph Laplaceans



Strong relation between $x^T L x$ and local distances between entries of x

➤ Let L = any matrix s.t. $L = D - W$, with $D = diag(d_i)$ and

$$w_{ij} \geq 0, \qquad d_i = \sum_{j \neq i} w_{ij}$$

Property 1: for any $x \in \mathbb{R}^n$:

$$x^\top L x = \frac{1}{2} \sum_{i,j} w_{ij} |x_i - x_j|^2$$

Property 2: (generalization) for any $Y \in \mathbb{R}^{d \times n}$:

$$\text{Tr}\,[Y L Y^\top] = \frac{1}{2} \sum_{i,j} w_{ij} \|y_i - y_j\|^2$$

Property 3: For the particular $L = I - \frac{1}{n} \mathbf{1}\mathbf{1}^\top$

$$X L X^\top = \bar{X} \bar{X}^\top == n \times \text{Covariance matrix}$$

Property 4: L is singular and admits the null vector e =`ones(n,1)`

Property 5: (Graph partitioning) Consider situation when $w_{ij} \in \{0, 1\}$. If x is a vector of signs (± 1) then

$$x^\top L x = 4 \times (\text{'number of edge cuts'})$$

edge-cut = pair (i, j) with $x_i \neq x_j$

➤ Would like to minimize (Lx, x) subject to $x \in \{-1, 1\}^n$ and $e^T x = 0$ [balanced sets]

➤ WIl solve a relaxed form of this problem

- Consider any symmetric (real) matrix A with eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ and eigenvectors $u_1, \cdots, u_n$

- Recall that:
(Min reached for $x = u_1$)

$$\min_{x\in\mathbb{R}^n} \frac{(Ax,x)}{(x,x)} = \lambda_1$$

- In addition:
(Min reached for $x = u_2$)

$$\min_{x\perp u_1} \frac{(Ax,x)}{(x,x)} = \lambda_2$$

- For a graph Laplacean $u_1 = e =$ vector of all ones and

- ...vector u_2 is called the Fiedler vector. It solves a relaxed form of the problem -

$$\min_{x\in\{-1,1\}^n;\ e^Tx=0} \frac{(Lx,x)}{(x,x)} \rightarrow \min_{x\in\mathbb{R}^n;\ e^Tx=0} \frac{(Lx,x)}{(x,x)}$$

- Define $v = u_2$ then $lab = sign(v - med(v))$