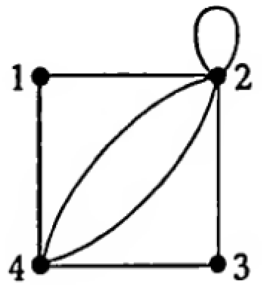
Graph
& its
Matrix Representation

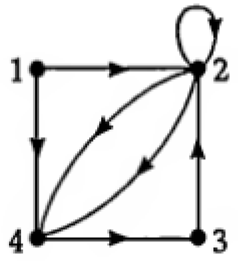
Matrix Representation

A graph is completely determined by either its adjacencies or its incidences (conveniently stated in matrix form)



$$\begin{array}{c|cccc} & \text{col } 1 & \text{col } 2 & \text{col } 3 & \text{col } 4 \\ \hline \text{row } 1 & 0 & 1 & 0 & 1 \\ \text{row } 2 & 1 & 1 & 1 & 2 \\ \text{row } 3 & 0 & 1 & 0 & 1 \\ \text{row } 4 & 1 & 2 & 1 & 0 \end{array}$$

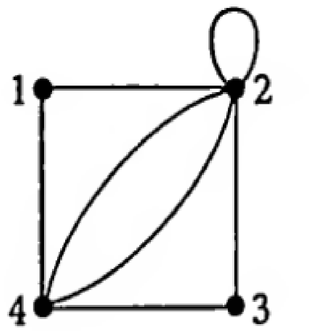
Adjacency matrix of G



$$\begin{array}{c|cccc} & \text{col } 1 & \text{col } 2 & \text{col } 3 & \text{col } 4 \\ \hline \text{row } 1 & 0 & 1 & 0 & 1 \\ \text{row } 2 & 0 & 1 & 0 & 2 \\ \text{row } 3 & 0 & 1 & 0 & 0 \\ \text{row } 4 & 0 & 0 & 1 & 0 \end{array}$$

Adjacency matrix of directed G

Matrix Representation



$$\begin{array}{c|cccc|}
 & \text{col } 1 & \text{col } 2 & \text{col } 3 & \text{col } 4 \\
 & \downarrow & \downarrow & \downarrow & \downarrow \\
\text{row } 1 \rightarrow & 0 & 1 & 0 & 1 \\
\text{row } 2 \rightarrow & 1 & 1 & 1 & 2 \\
\text{row } 3 \rightarrow & 0 & 1 & 0 & 1 \\
\text{row } 4 \rightarrow & 1 & 2 & 1 & 0
\end{array}$$

A labeled graph and its adjacency matrix

- The adjacency matrix of a graph is symmetrical about the main diagonal (top-left to bottom-right)

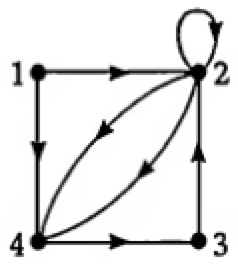
- Also, for a graph without loops, each entry on the main diagonal is 0

- The sum of the entries in any row or column is the degree of the vertex corresponding to that row or column

Definition

Let G be a graph with n vertices labeled 1,2,3, ..., n.
The adjacency matrix A(G) of G is the n x n matrix in which the entry in row i and column j is the number of edges joining the vertices i and j.

Matrix Representation



$$\begin{array}{c|cccc|}
 & \text{col } 1 \downarrow & \text{col } 2 \downarrow & \text{col } 3 \downarrow & \text{col } 4 \downarrow \\
\text{row } 1 \rightarrow & 0 & 1 & 0 & 1 \\
\text{row } 2 \rightarrow & 0 & 1 & 0 & 2 \\
\text{row } 3 \rightarrow & 0 & 1 & 0 & 0 \\
\text{row } 4 \rightarrow & 0 & 0 & 1 & 0
\end{array}$$

Adjacency matrix of a labeled digraph G

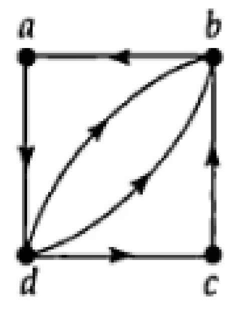
- Not usually symmetrical about the main diagonal
- If the digraph has no loops, then each entry on the main diagonal is 0
- The sum of the entries in any row is the out-degree of the vertex corresponding to that row
- The sum of the numbers in any column is the in-degree of the vertex corresponding to that column

Definition

Let D be a digraph with n vertices labelled.1, 2, 3, ...,J.
The adjacency matrix A(D) of D is the n x n matrix in which the entry in row i and column j is the number of arcs from vertex i to vertex j.

Matrix Representation

Adjacency matrix of a labeled digraph G



$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 0 & 0 & 1 \\ b & 1 & 0 & 0 & 0 \\ c & 0 & 1 & 0 & 0 \\ d & 0 & 2 & 1 & 0 \end{array}$$

Walks of lengths 2:
For example, there are two different walks of length 2 from a to b, because there is one arc from a to d and two arcs from d to b.

Walks of lengths 3:
Similarly, there are two different walks of length 3 from d to d, since there are two arcs from d to b, and one walk of length 2 from b to d, namely, bad.

Walks of lengths 1:
the number of walks of length 1 from a to c is 0, so 0 appears in row 1 column 3;
the number of walks of length 1 from b to a is 1, so 1 appears in row 2 column 1;
the number of walks of length 1 from d to b is 2, so 2 appears in row 4 column 2.

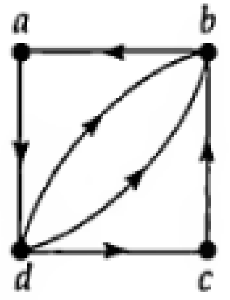
	a	b	c	d
a		2		
b				1
c				
d				

numbers of walks of length 2

	a	b	c	d
a				
b				
c				
d				2

numbers of walks of length 3

- Complete the matrices…



	a	b	c	d
a	0	0	0	1
b	1	0	0	0
c	0	1	0	0
d	0	2	1	0

	a	b	c	d
a		2		
b				1
c				
d				

numbers of walks of length 2

	a	b	c	d
a				
b				
c				
d				2

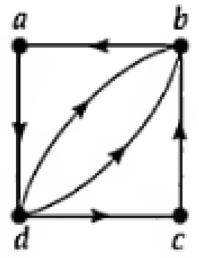
numbers of walks of length 3

- Complete the matrices…

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}, \quad A^2 = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}, \quad A^3 = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix}.$$

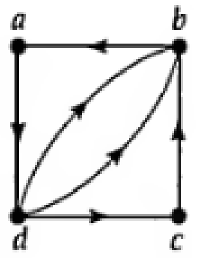
Let G be a digraph with n vertices labeled 1,2, ..., n, let A be its adjacency matrix with respect to this listing of the vertices, and let k be any positive integer. Then the number of walks of length k from vertex i to vertex j is equal to the entry in row i and column j of the matrix A^k (the k^th power of the matrix A)

Matrix Representation



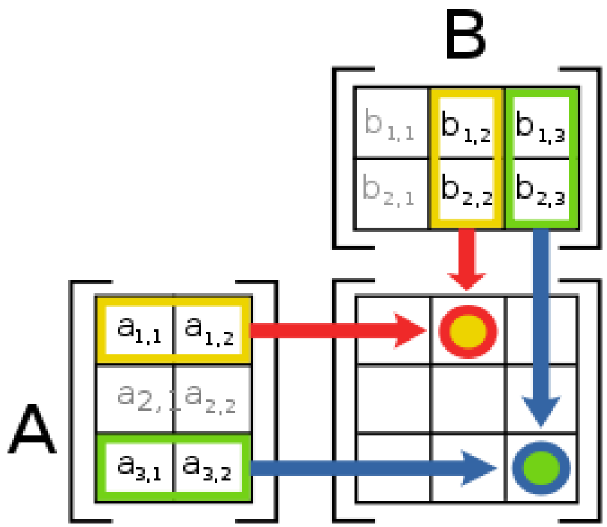
$$\begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 0 & 0 & 1 \\ b & 1 & 0 & 0 & 0 \\ c & 0 & 1 & 0 & 0 \\ d & 0 & 2 & 1 & 0 \end{array}$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$

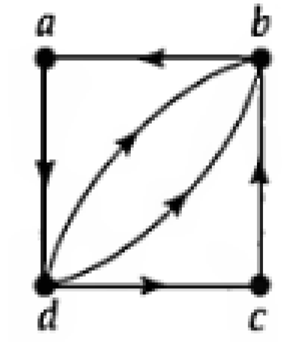
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}$$



$$\begin{aligned} x_{1,2} &= (a_{1,1}, a_{1,2}) \cdot (b_{1,2}, b_{2,2}) \\ &= a_{1,1}b_{1,2} + a_{1,2}b_{2,2} \\ x_{3,3} &= (a_{3,1}, a_{3,2}) \cdot (b_{1,3}, b_{2,3}) \\ &= a_{3,1}b_{1,3} + a_{3,2}b_{2,3}. \end{aligned}$$

Matrix Representation

A digraph is strongly connected if there is a path from vertex i to vertex j, for each pair of distinct vertices i and j, and that a path is a walk in which all the vertices are different.

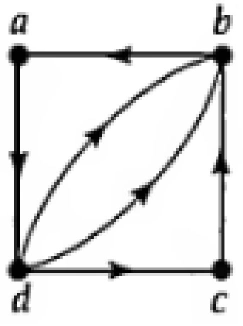


There are four vertices, so a path has length 1,2 or 3.

The numbers of walks (including the paths) of lengths 1, 2 and 3 between pairs of distinct vertices are given by the non-diagonal entries in the matrices in
A, A^2, and A^3

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}, \quad A^2 = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}, \quad A^3 = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix}.$$

Matrix Representation



$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \end{bmatrix}, \quad A^2 = \begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}, \quad A^3 = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 \end{bmatrix}.$$

$$B = A + A^2 + A^3 = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 3 & 3 & 1 & 2 \end{bmatrix}.$$

Let b_{ij} denote the entry in row i and column j in the matrix B. Then each entry b_{ij} is the total number of walks of lengths l, 2 and 3 from vertex i to vertex j.
Since all the non-diagonal entries are positive, each pair of distinct vertices is connected by a path, so the digraph is strongly connected.

Theorem: Let D be a digraph with n vertices labeled 1,2, ...,n, let A be its adjacency matrix with respect to this listing of the vertices, and let B be the matrix

$$B = A + A^2 + \dots + A^{n-1}$$

Then D is strongly connected iff each non-diagonal entry in B is positive
- that is, $B_{ij} > 0$ whenever $i \neq j$

Pros and Cons of Adjacency Matrices

- Pros:
  - Simple to implement
  - Easy and fast to inform if a pair (i,j) is an edge: simply check if A[i][j] is 1 or 0
- Cons:
  - No matter how few edges the graph has, the matrix takes $O(n^2)$ in memory

Adjacency Lists Representation

- A graph of n nodes is represented by a one-dimensional array L of linked lists, where
  - L[i] is the linked list containing all the nodes adjacent from node i
  - The nodes in the list L[i] are in no particular order

Example of Linked Representation

L[0]: empty

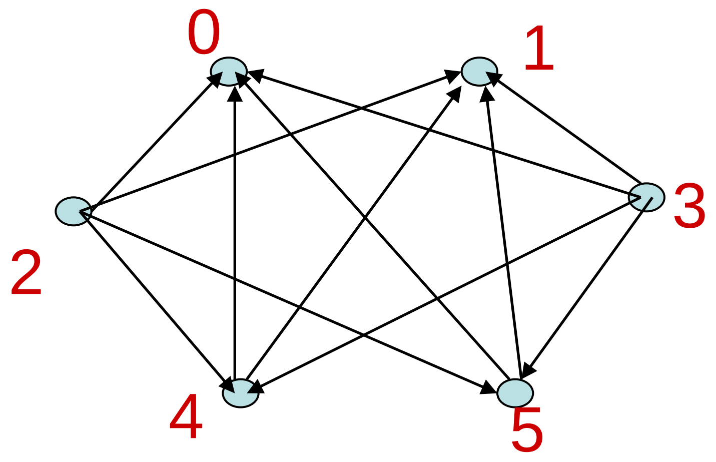
L[1]: empty

L[2]: 0, 1, 4, 5

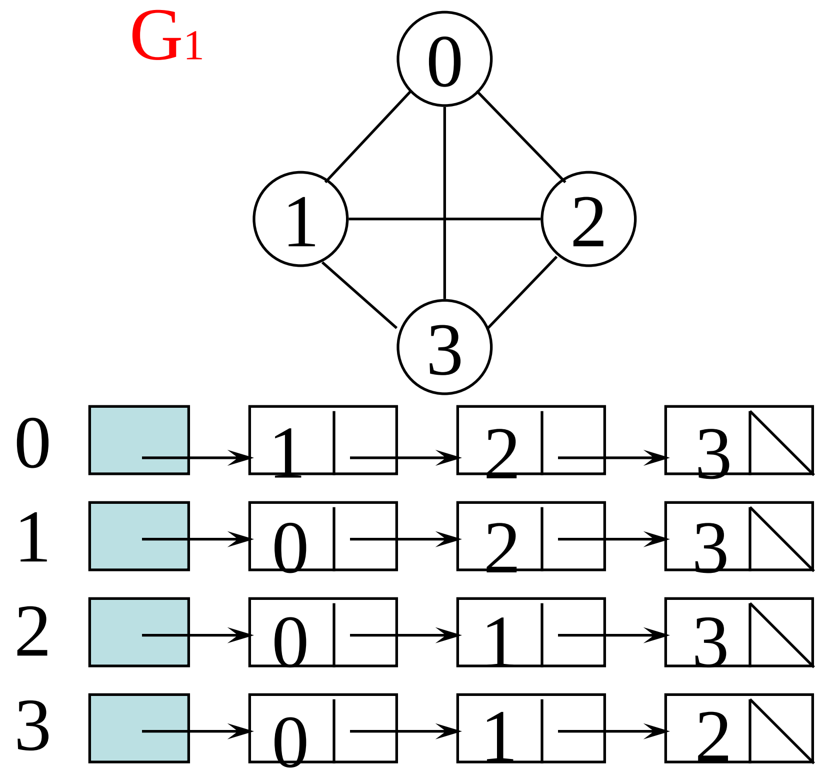
L[3]: 0, 1, 4, 5

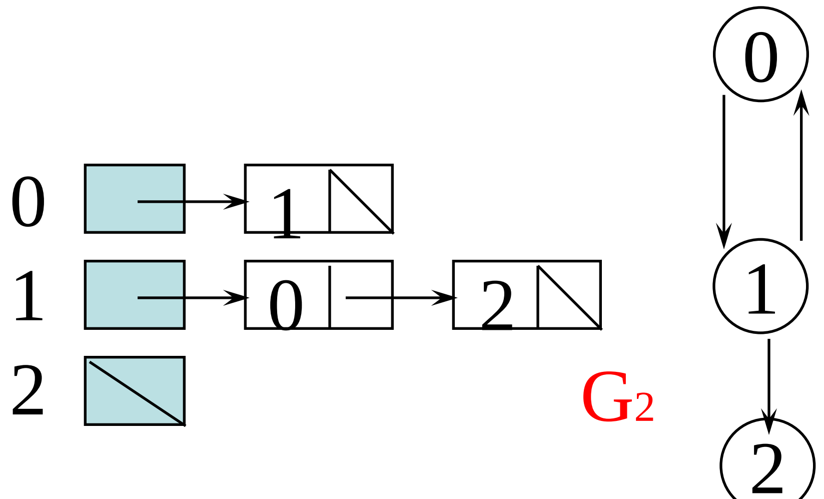
L[4]: 0, 1

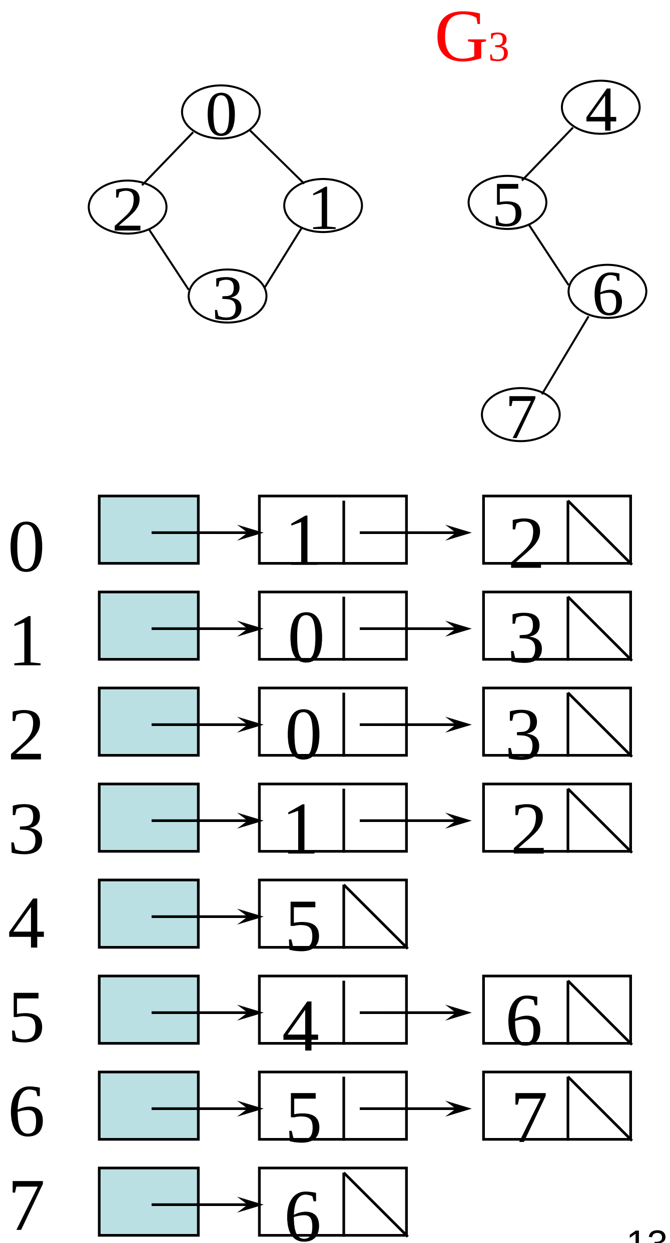
L[5]: 0, 1



Examples of Adjacency Lists







Pros and Cons of Adjacency Lists

- Pros:
  - Saves on space (memory): the representation takes as many memory words as there are nodes and edges
- Cons:
  - It can take up to O(n) time to determine if a pair of nodes (i,j) is an edge: one would have to search the linked list L[i], which takes time proportional to the length of L[i].

The Graph Class

```
class Graph {
  public:
     typedef int  datatype;
     typedef datatype * datatypeptr;
     Graph( int n=0); // creates a graph of n nodes and no edges
     bool isEdge( int i, int j);
     void setEdge( int i, int j, datatype x);
     int getNumberOfNodes(){return numberOfNodes;};
private:
     datatypeptr *p;   //a 2-D array, i.e.an adjacency matrix
     int numberOfNodes;
};
```

Graph Class Implementation

```
Graph::Graph( int n){
    assert(n>=0);
    numberOfNodes=n;
    if (n==0)   p=NULL;
    else{
       p = new datatypeptr[n];
       for (int i=0;i<n;i++){
           p[i] = new datatype[n];
           for (int j=0;j<n;j++)
               p[i][j]=0;
       }
    }
};
```

```
bool Graph::isEdge(int i, int j){
            assert(i>=0 && j>=0);
            return p[i][j] != 0;
};
```

```
void Graph::;setEdge(int i,
                    int j, datatype x){
            assert(i>=0 && j>=0);
            p[i][j]=x;
};
```

Theorem:

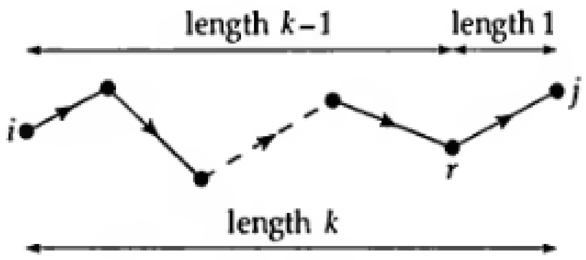
Let D be a digraph with n vertices labeled 1,2, ..., n, let A be its adjacency matrix with respect to this listing of the vertices, and let k be any positive integer.
Then the number of walks of length k from vertex i to vertex j is equal to the entry in row i and column j of the matrix A^k (the kth power of the matrix A).

Proof:

The proof is by mathematical induction on k, the length of the walk.

Step 1 The statement is true when k = 1, since the number of walks of length 1 from vertex i to vertex j is the number of arcs from vertex i to vertex j, and this is equal to a_{ij} the entry in row i and column j of the adjacency matrix A.

Step 2 We assume that k > 1, and that the statement is true for all positive integers less than k. We wish to prove that the statement is true for the positive integer k.



Consider any walk of length k from vertex i to vertex j. Such a walk consists of a walk of length k - 1 from vertex i to some vertex r adjacent to vertex j, followed by a walk of length 1 from vertex r to vertex j.

By our assumption, the number of walks of length k-1 from vertex i to vertex r is the entry in row i and column r of the matrix A^{k-1}, which we denote by $a_{ir}^{(k-1)}$.

Since the number of walks of length 1 from vertex r to vertex j is a_{rj} it follows that

the number of walks of length k from vertex i to vertex j via vertex r (at the

previous step) = a_{ir}^{k-1} £ a_{rj}

So, the total number of walks of length k from vertex i to vertex j

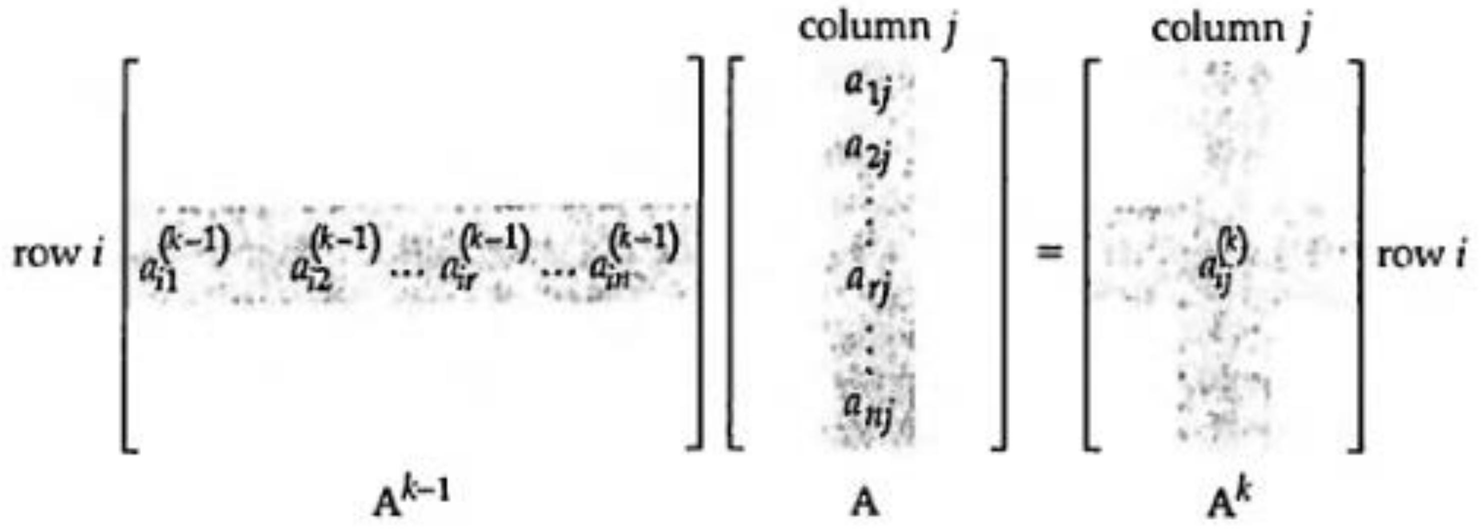
=

the number of such walks via vertex 1 (at the previous step)
+ the number of such walks via vertex 2 (at the previous step)
…….

+ the number of such walks via vertex r (at the previous step)
+ the number of such walks via vertex n (at the previous step),

$$a_{i1}^{(k-1)}a_{1j}+a_{i2}^{(k-1)}a_{2j}+\dots+a_{ir}^{(k-1)}a_{rj}+\dots+a_{in}^{(k-1)}a_{nj}$$

By the rules for matrix multiplication, this is just the entry in row i and column j of the matrix $A^{k-1} A = A^k$, as required.



Thus, if the statement is true for all positive integers less than k, then it is true for the integer k. This completes Step 2.

Therefore, by the principle of mathematical induction, the statement is true for all positive integers k. .

Theorem

Let D be a digraph with n vertices labeled 1,2, ...,n, let A be its adjacency matrix with respect to this listing of the vertices, and let B be the matrix

$$B= A+A^2+ \dots +A^{n-1} .$$

Then D is strongly connected if and only if each non-diagonal entry in B is positive - that is, $b_{ij} > 0$ whenever $i \neq j$.

Proof

There are two statements to prove.

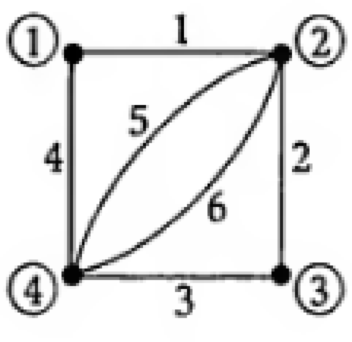
(a) If each non-diagonal entry in B is positive, then D is strongly connected.

Let D be a digraph that satisfies the given conditions, and suppose that each non-diagonal entry in B is positive - that is, bij > 0 whenever $i \neq j$ - then $A_{ij}^{(k)} > 0$ for some $k \cdot n - 1$. Therefore there is a walk of length at most n -1 from vertex i to vertex j whenever $i \neq j$, so the digraph D is strongly connected.

(b) If the digraph D is strongly collected, then each no-diagonal entry in B is positive.

Let D be a strongly connected digraph that satisfies the given conditions; then there is a path from any vertex to any other. Since D has n vertices, such a path has length at most n - 1. It follows that $a_{ij}^{(k)} > 0$ for at least one value of $k \cdot n-1$, and hence that the entry in row i and column j of B is positive; that is, $b_{ij} > 0$ whenever $i \neq$ j. .

Incidence Matrix



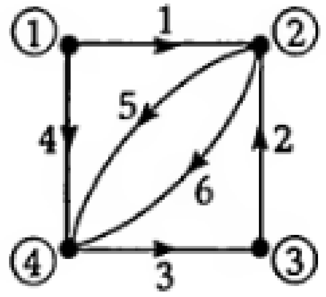
$$\begin{array}{c|cccccc} & \text{col } 1 & \text{col } 2 & \text{col } 3 & \text{col } 4 & \text{col } 5 & \text{col } 6 \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \hline \text{row } ① \rightarrow & 1 & 0 & 0 & 1 & 0 & 0 \\ \text{row } ② \rightarrow & 1 & 1 & 0 & 0 & 1 & 1 \\ \text{row } ③ \rightarrow & 0 & 1 & 1 & 0 & 0 & 0 \\ \text{row } ④ \rightarrow & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Definition

Let G be a graph without loops, with n vertices labeled $u_1, u_2, \ldots u_n$, and m edges labeled $e_1, e_2, \ldots e_m$.

The incidence matrix I(G) of G is the n x m matrix in which the entry in row i and column j is

1 if the vertex i is incident with the edge j,
0 otherwise.



$$\begin{array}{c|cccccc} & \text{col } 1 \downarrow & \text{col } 2 \downarrow & \text{col } 3 \downarrow & \text{col } 4 \downarrow & \text{col } 5 \downarrow & \text{col } 6 \downarrow \\ \hline \text{row } ① \rightarrow & 1 & 0 & 0 & 1 & 0 & 0 \\ \text{row } ② \rightarrow & -1 & -1 & 0 & 0 & 1 & 1 \\ \text{row } ③ \rightarrow & 0 & 1 & -1 & 0 & 0 & 0 \\ \text{row } ④ \rightarrow & 0 & 0 & 1 & -1 & -1 & -1 \end{array}$$

In the incidence matrix of a digraph without loops, each column has exactly one 1 and one -1, since each arc is incident from one vertex and incident to one vertex;

the number of 1's in any row is the out-degree of the vertex corre-sponding to that row, and

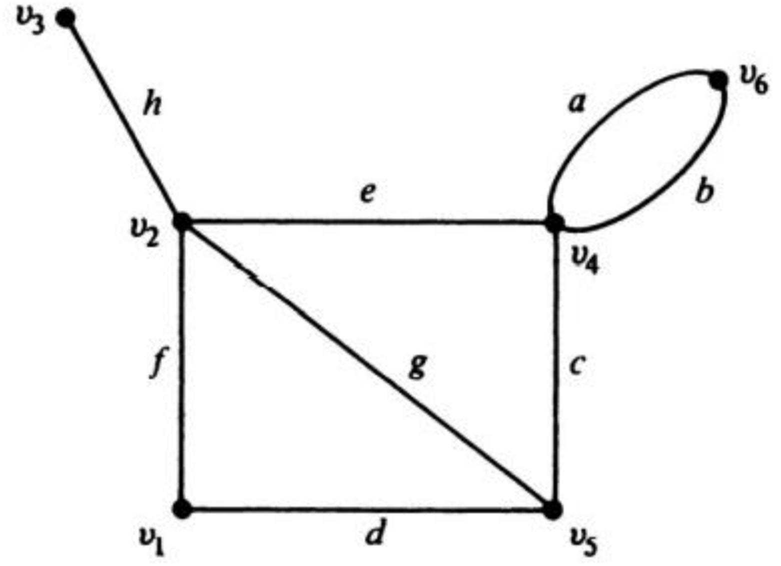
the number of -1's in any row is the in-degree of the vertex corresponding to that row.

Definition

Let D be a digraph without loops, with n vertices labeled u_1,u_2, .., u_n and m arcs labeled e_1, e_2, … e_m.

The incidence matrix I(D) of D is the n x m matrix in which the entry in row; and column j is

1 if arc j is incident from vertex;
-1 if arc j is incident to vertex;
0 otherwise.



Find Incidence Matrix of the Graph…

$$
\begin{array}{c|cccccccc}
 & a & b & c & d & e & f & g & h \\
\hline
v_1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
v_2 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
v_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
v_4 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
v_5 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
v_6 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

- Parallel edges in a graph produce identical columns in its incidence matrix, e.g.columns 1&2
- Permutation of any two rows or columns in an incidence matrix simply corresponds to relabeling the vertices and edges of the same graph

→ Two graphs G1 and G2 are isomorphic iff their incidence matrices differ only by permutations of rows and columns

- If a graph G is **disconnected** and consists of two components g1 and g2 , the incidence matrix A(G) of graph G can be written in a block-diagonal form as:

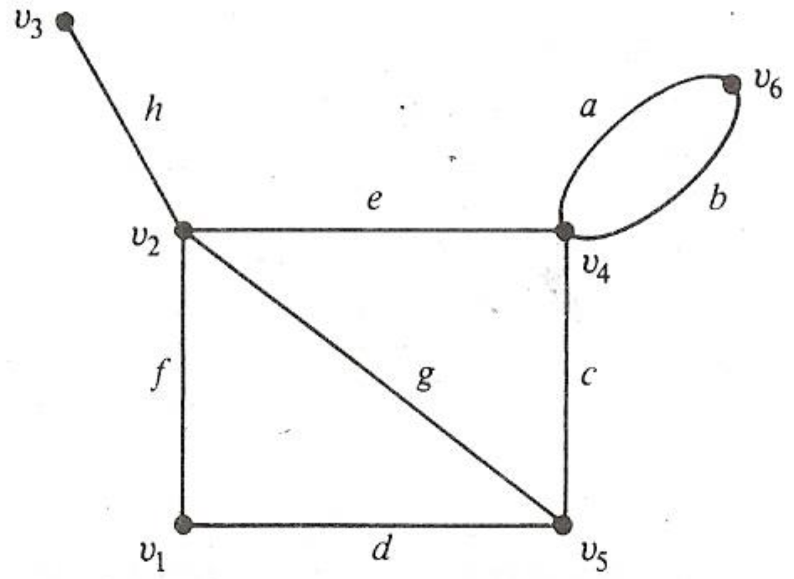
$$A(G) = \left[\begin{array}{c|c} A(g_1) & 0 \\ \hline 0 & A(g_2) \end{array}\right]$$

Where, A(g1) and A(g2) are the incidence matrices of components g1 and g2

→ for a disconnected graph with any number of components;
no edge in g1 is incident on vertices of g2 , *and vice a versa*
→ square submatrix of A(G) is **nonsingular** iff the corresponding subgraph is a tree. The tree in this case is a **spanning tree**, because it contains n − 1 edges of the n-vertex graph

*// A **non-singular matrix** is a square one whose determinant is not zero //*

Circuit Matrix



If Graph G contains q different circuits and e edges then circuit matrix B is of size q*e

b_{ij} =1 if ith circuit includes jth edge
=0 otherwise

$$B(G) = \begin{array}{c|cccccccc} & a & b & c & d & e & f & g & h \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 4 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{array}.$$

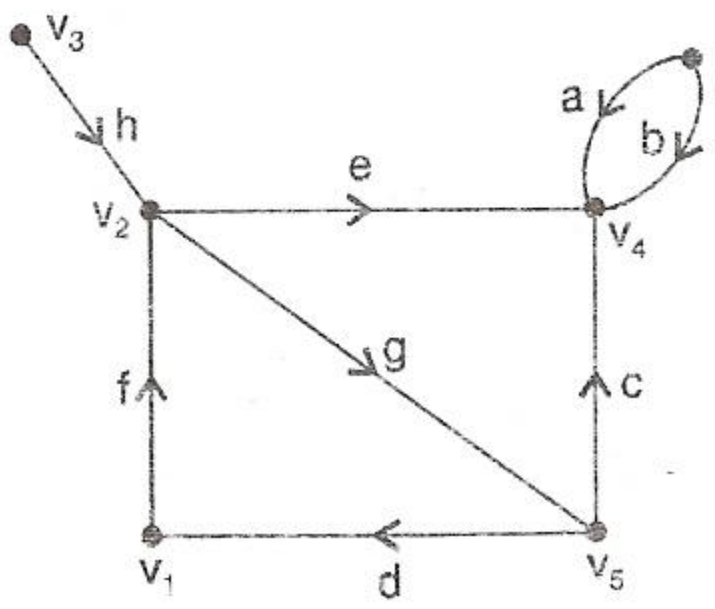
- Let B(G) has 4 different circuits.

1:{a,b} 2:{c,e,g}

3:{d,f,g} 4:{c,d,f,e}

- Two graphs G1 & G2 have same circuit matrices iff G1 & G2 are 2-isomorphic (obtained after splitting & separation)

Circuit Matrix of a Digraph



b_{ij} =1 if ith circuit includes jth edge & orintation of the edge and circuit coincide

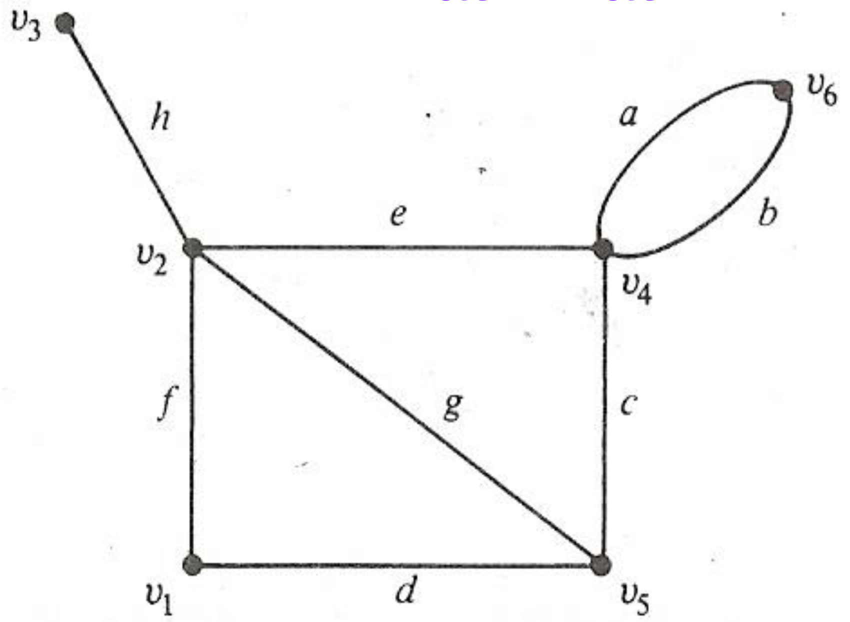
= -1 if ith circuit includes jth edge but the orientations of the two are opposite

= 0 otherwise

(circuit does not include edge)

$$\begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Path Matrix



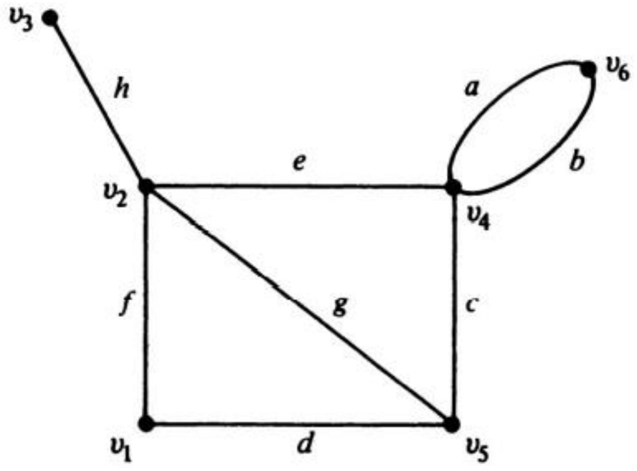
- A path matrix P(x,y) indicates path between pair of vertices x & y
- The rows in P(x,y) corresponds to different paths in vertex x & y
- The columns correspond to the edges in G & P(x,y) = $[p_{i,j}]$
- $p_{i,j}$= 1 if jth edge lies in the ith path

  = 0 otherwise

- For P(v3,v4) 3 different path exist
- Let path 1: {h,e} 2: {h,g,c} & 3: {h,f,d,c}

$$P(v_3, v_4) = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

- The ring sum of any two rows in P(x,y) corresponds to a circuit or an edge-disjoint union of circuits.



- Let B and A be, respectively, the circuit matrix and the incidence matrix (of a self-loop-free graph) whose columns are arranged using the same order of edges. Then every row of B is **orthogonal** to every row A; that is,

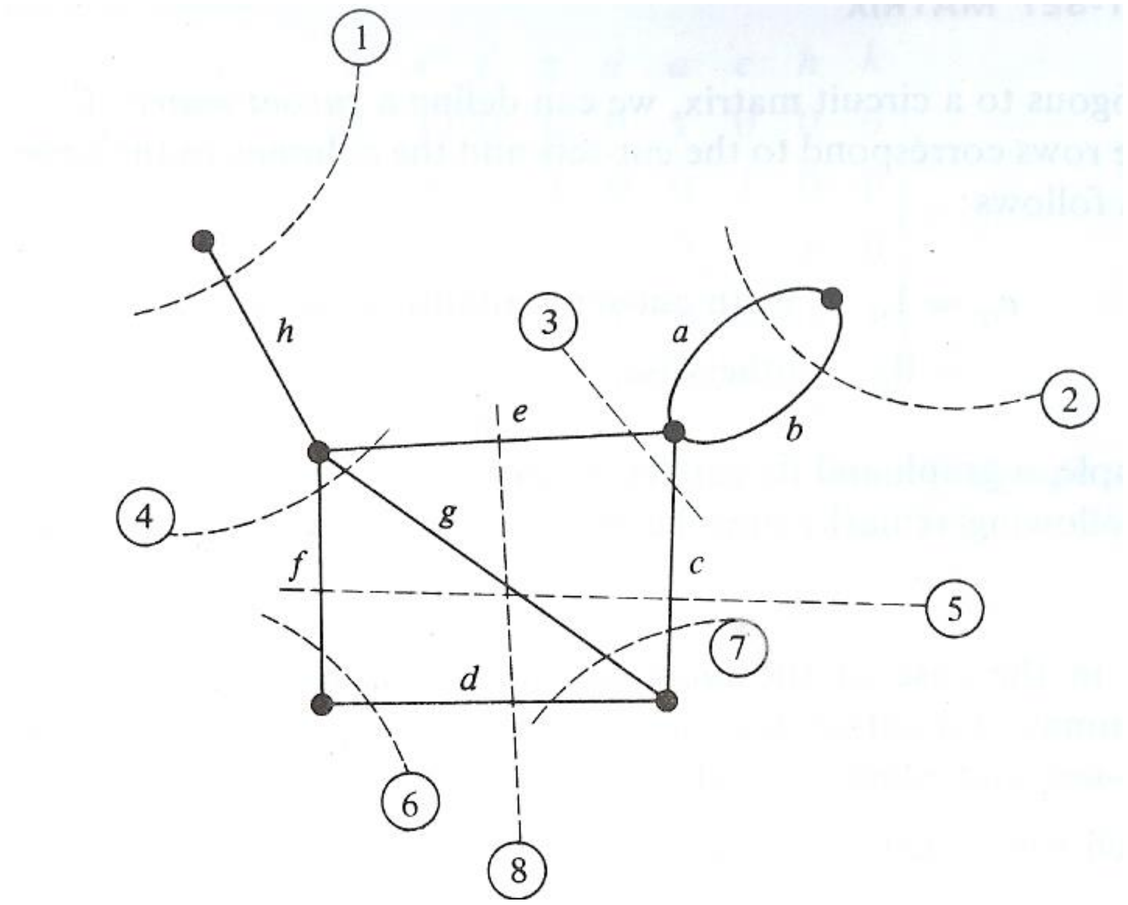
$$A \cdot B^T = B \cdot A^T = 0 \qquad (\text{mod } 2).$$

$$A \cdot B^T = \begin{bmatrix} 0&0&0&1&0&1&0&0 \\ 0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&1 \\ 1&1&1&0&1&0&0&0 \\ 0&0&1&1&0&0&1&0 \\ 1&1&0&0&0&0&0&0 \end{bmatrix} \cdot \begin{bmatrix} 1&0&0&0 \\ 1&0&0&0 \\ 0&1&0&1 \\ 0&0&1&1 \\ 0&1&0&1 \\ 0&0&1&1 \\ 0&1&1&0 \\ 0&0&0&0 \end{bmatrix}$$

$$= \begin{bmatrix} 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \\ 0&0&0&0 \end{bmatrix} \qquad (\text{mod } 2).$$

Multiply the incidence matrix and transposed circuit of the graph making sure that the edges are in the same order in both

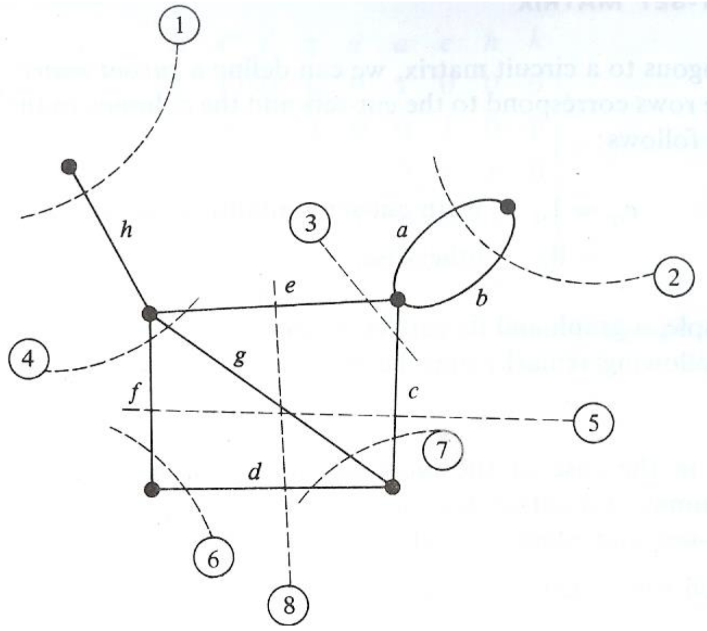
Cut-set Matrix



- C=[$c_{i,j}$]; Rows correspond to cut-sets; columns to the edges
- $c_{i,j}$ =1 if ith cut-set contains j^{th} edge

  = 0 otherwise

Cut-set Matrix



$$C = \begin{array}{c|cccccccc} & a & b & c & d & e & f & g & h \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 5 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 6 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 7 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 8 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}$$

Significance

Why Graphs in Matrix Representations?

- Handling Large computational problems
- Practically impossible for hand computation
- Programming attempts to handle problems like PERT, flow problems, transportation n/w, electrical n/w, circuit n/w etc.

e.g. Travelling Salesman Problem

Travelling Salesman Problem_Analysis

- Problem of finding lowest-weight Hamiltonian Circuit : weighted complete graph with n vertices
- There are (½ * (n-1)!) different circuits so as solutions.
- Brute Force analysis for 10 vertex graph:

Total circuits= (½ * (n-1)!) = (½ * (10-1)!) = 181,440

- For 20 vertex graph:

Total circuits= (½ * (n-1)!) = (½ * (20-1)!) $\approx$ 6* 10^{16}

- Even with nano instructional operations time required $\approx$ 2 years

Analysis

- Manipulation & analysis of graphs is essentially non_numerical
- Graph Theoretic algorithms & programs use decision making ability of the computer rather than ability to perform arithmetic operations
- Features of Algorithm:
  - Finiteness
  - Definiteness
  - i/p
  - o/p
  - effectiveness

Efficiency of Algorithms

- Efficiency in terms of memory & computation time
- Function of the size of the input
- Input is graph
- Size: f(n, e)
- Evaluation with "Worst case" execution time

INPUT: Computer Representation of a Graph

A) Adjacency Matrix:

- Most popular form
- n x n binary matrix during i/p, process & o/p
- Hence require n^2 bits
- Each row of matrix with $\lceil n/w \rceil$ m/c words (w is the word length)
- Total number of words = $n \times \lceil n/w \rceil$
- for symmetric matrix bits for storage = $n(n-1)/2$ but increases computation time

INPUT:
Computer Representation of a Graph

B) Incidence Matrix:

- Uses n.e bits of storage
- Normally e > n : hence $n.e > n^2$
- Favored in electrical & switching n/w

INPUT:
Computer Representation of a Graph

C) Edge Listing: (e.g. diagraph)

- Arbitrary order can be allowed
- Parallel edges & loops can be represented unlike adjacency matrix
- The number of bits required to label each vertex is b :

  where ($2^{b-1} < n <= 2^b$)
- Each edge requires storing two vertices; hence total storage

  = 2e.b bits
- More economical than adjacency matrix if : $2e.b < n^2$
- If adjacency matrix of a graph is *sparse*; edge listing is more efficient approach *(A mat. with many 0 elements is called as sparse mat. A` sparse adjacency mat. Implies a small e/n ratio)*
- However, edge listing input style cause difficulty (extensive search techniques) in storage, retrieval & manipulation of the graph

INPUT:
Computer Representation of a Graph

D) Two Linear Arrays:

- Slight variation in edge listing
- Graph with two linear arrays:

  F=(f1, f2, ...fe) & H=(h1, h2, ...he)

- *e.g.*
- The *i*th edge *ei* is from vertex *fi* to vertex *hi*
- Convenient sorting in weighted graphs
- Storage requirement is same as edge listing

INPUT: Computer Representation of a Graph

E) Successor Listing:

- Efficient method in graphs in which ratio e/n is not large is by means of n linear arrays
- vertices can be arranged in any order 1, 2, .. n
- Each vertex k by linear array, whose first element is k & remaining are the successors of k
- Let d_{av} be the average out degree of the graph; if one word per vertex;

  total storage for n vertex graph is: $n(1+ d_{av})$ words
- Successor listing is more efficient than adjacency mat. If :

  $$(d_{av} < (\lceil n/w \rceil) - 1)$$
- This form of the i/p is extremely convenient for path finding algorithms & for depth first search of a graph.

Comment….forward..

- Graph representation forms are not entirely different; but coveys same information
- Efficiency of algorithm is dependant on the form of the i/p
- Proper choice of the data structure is important
- The o/p varies with problem (giving o/p in the form of sub_graph, yes/no, distance… etc…)