Graph Representation

What is Graph ?

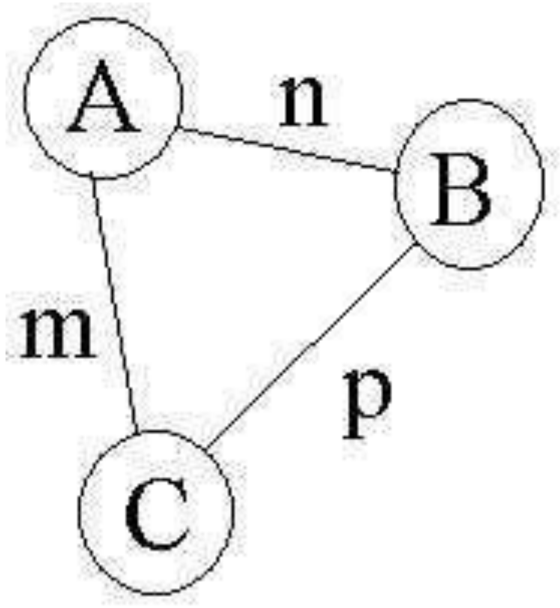
A graph G consists of a finite set of ordered pairs, called **edges E**, of certain entities called **vertices V**.

Edges are also called as arcs or links.

Vertices are also called as nodes or points.

$$G=(V,E)$$

- A graph is a set of vertices and edges. A vertex may represent a state or a condition while the edge may represent a relation between two vertices.


A
B
C
n
m
p

Types of Representation

Two ways are there for representing graph in the memory of a computer.

They are:-

- Sequential Representation.
- Linked Representation.

Sequential Representation

Graphs can be represented through matrix in system's memory. This is sequential in nature. This type of representation is called sequential representation of graphs.

Types of Sequential Representation

- Adjacency Matrix Representation.
- Incidence Matrix Representation.
- Circuit Matrix Representation.
- Cut Set Matrix Representation.
- Path Matrix Representation.

Linked Representation

- Graphs can be represented through Linked List in system's memory. This is Linked in nature. This type of representation is called Linked representation of graphs.

Types of Linked Representation

- Adjacency List Representation.

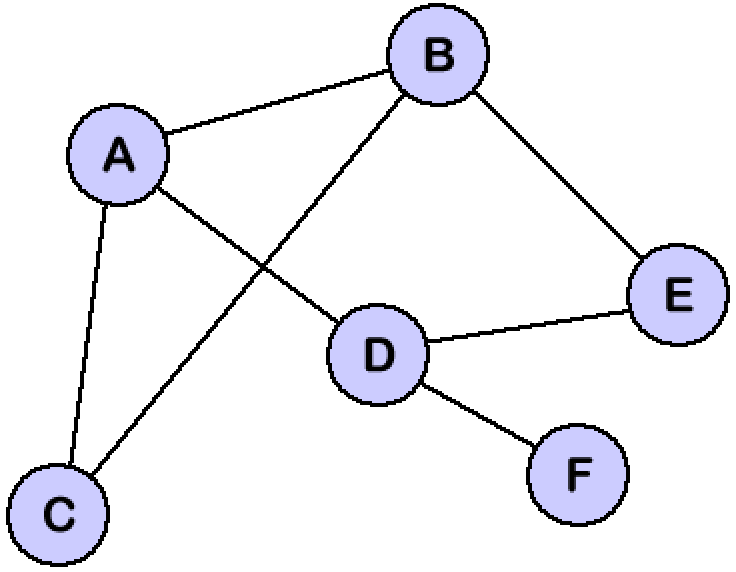
Adjacency Matrix Representation

The Adjacency matrix of a graph G with n vertices is N x N. It is given by A=[a_{ij}].

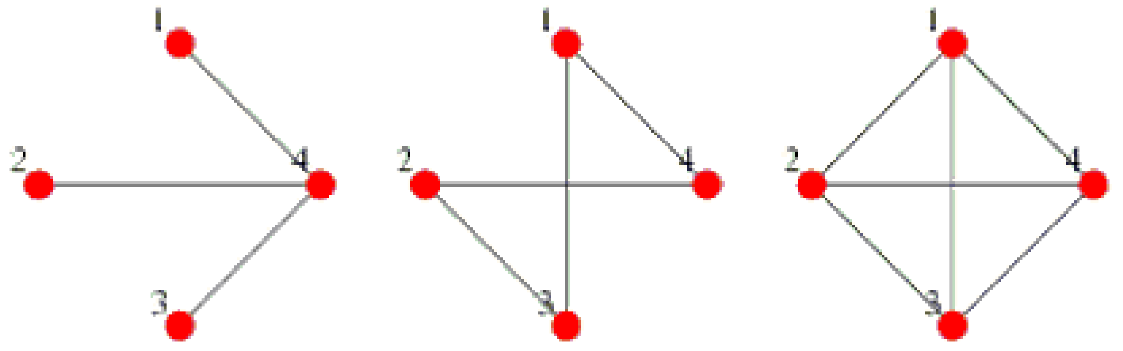
a_{ij} =1 if ith and jth vertices are adjacent.

=0 if ith and jth vertices are not adjacent.

Examples



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



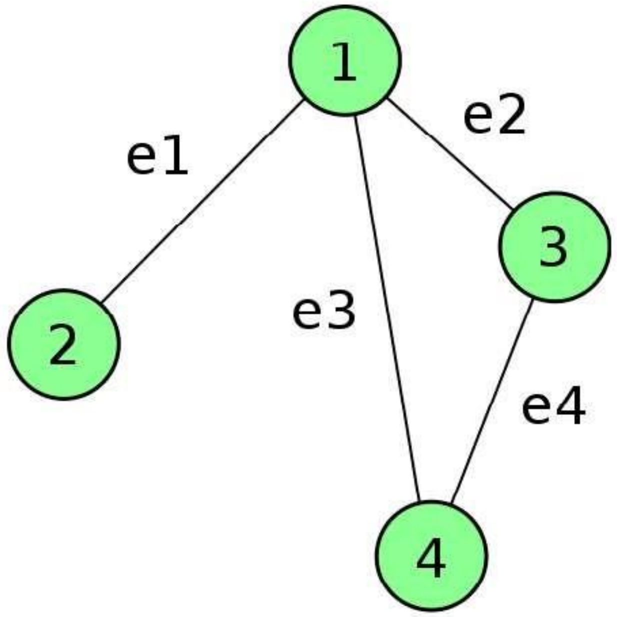
$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Incidence Matrix Representation

- The Incidence matrix of a graph G with N vertices and E edges is N x E.

$$m_{ij} = 1 \text{ if } e_j \text{ is incident on } v_i$$
$$= 0 \text{ otherwise}$$

Examples



$$
\begin{array}{c c}
 & \begin{array}{cccc} e1 & e2 & e3 & e4 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} & \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}
\end{array}
$$

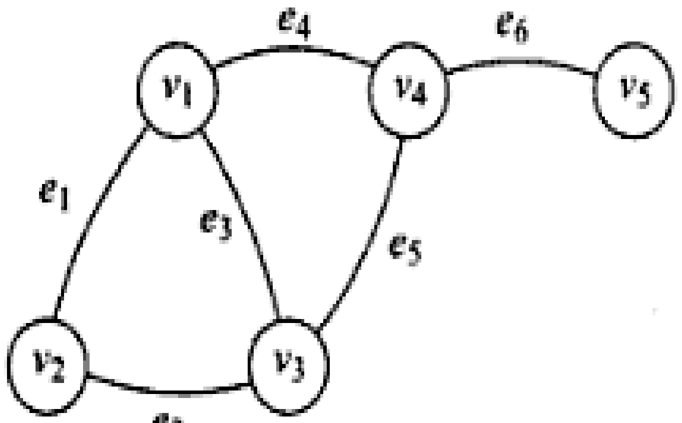
Circuit Matrix Representation

- Circuit Matrix is represented by the number of different circuits T and the number of edges E is T x E in the graph. The Circuit Matrix $C=C_{ij.}$

  Cij=1 if i^{th} circuit includes j^{th} edge.

  = 0 otherwise.

Examples



(a) Graph G_{14}

$$C = \begin{array}{c|cccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \hline 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 1 & 1 & 0 \\ 3 & 1 & 1 & 0 & 1 & 1 & 0 \end{array}$$

Circuit 1 : $\{e_1, e_2, e_3\}$
Circuit 2 : $\{e_3, e_4, e_5\}$
Circuit 3 : $\{e_1, e_2, e_5, e_4\}$

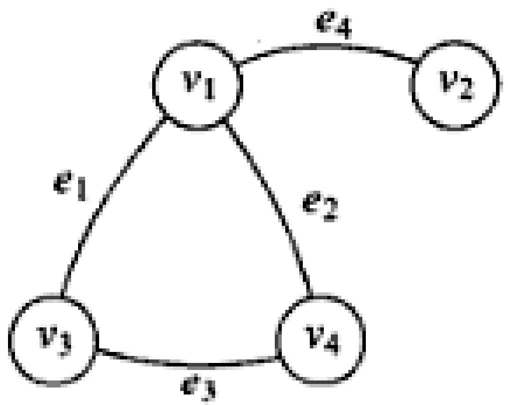
(b) Circuit matrix of G_{14}

Cut Set Matrix Representation

- A Matrix $S = S_{ij}$ Rows correspond to cut sets and columns correspond to edges of the graph is defined to be a cut set matrix.

S_{ij} = 1 if the i^{th} cut set contains the j^{th} edge
= 0 otherwise

Examples



(a) Graph G_{15}

$$S = \begin{array}{c|cccc|l} & e_1 & e_2 & e_3 & e_4 & \\ 1 & 0 & 0 & 0 & 1 & 1 : \{e_4\} \\ 2 & 1 & 1 & 0 & 0 & 2 : \{e_1, e_2\} \\ 3 & 0 & 1 & 1 & 0 & 3 : \{e_2, e_3\} \\ 4 & 1 & 0 & 1 & 0 & 4 : \{e_1, e_3\} \end{array}$$

(b) Cut set matrix of G_{15}

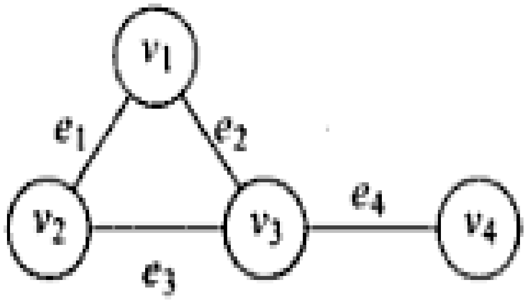
Path Matrix Representation

- A path matrix is generally defined for a specific pair of vertices then the path matrix denoted as $P(u,v) = P_{ij}$.

P_{ij} = 1 if the jth edge lies in the ith path.

= 0 otherwise.

Examples



(a) Graph G_{16}

$$P(v_1, v_4) = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ \hline 1 & 0 & 1 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 \end{array}$$

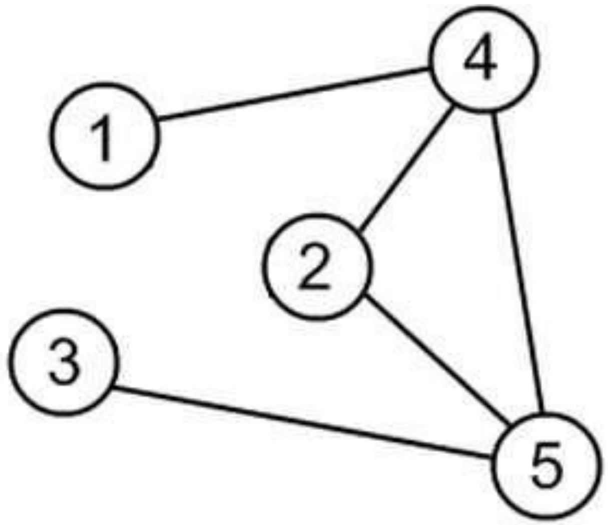
Paths :
1 : $\{e_2, e_4\}$
2 : $\{e_1, e_3, e_4\}$

(b) Path matrix between v_1 v_4 of G_{16}

Adjacency List representation

- Adjacency List representation is one of the alternatives to adjacency matrix. It requires less amount of memory. For every vertex adjacency list stores a list of vertices, which are adjacent to current one.

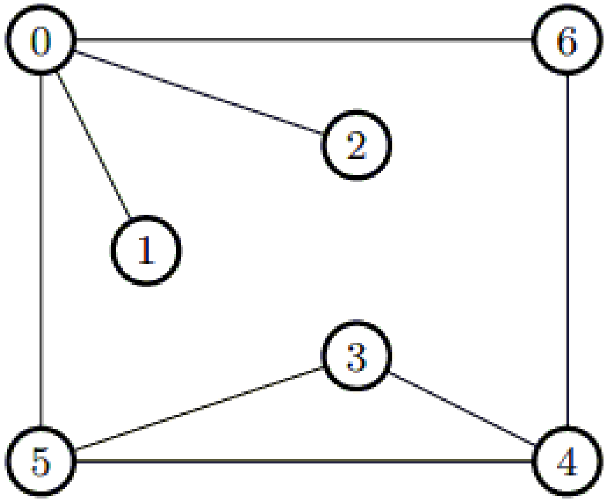
Examples

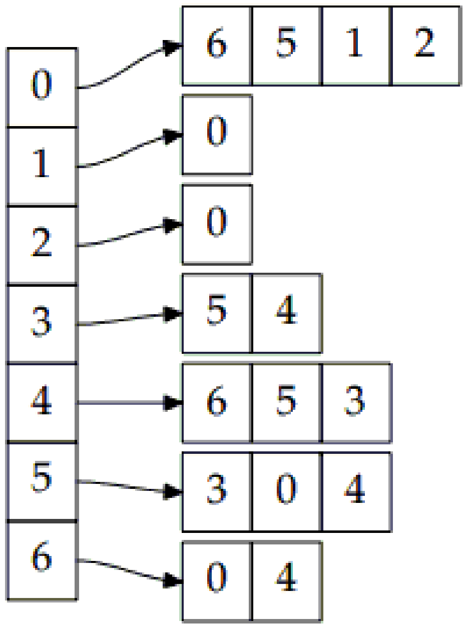


Graph

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

Adjacency list


0
6
2
1
3
5
4


0
1
2
3
4
5
6
6 5 1 2
0
0
5 4
6 5 3
3 0 4
0 4

In Computer’s

- Even there are many mathematical representations…

  adjacency matrix

  and adjacency

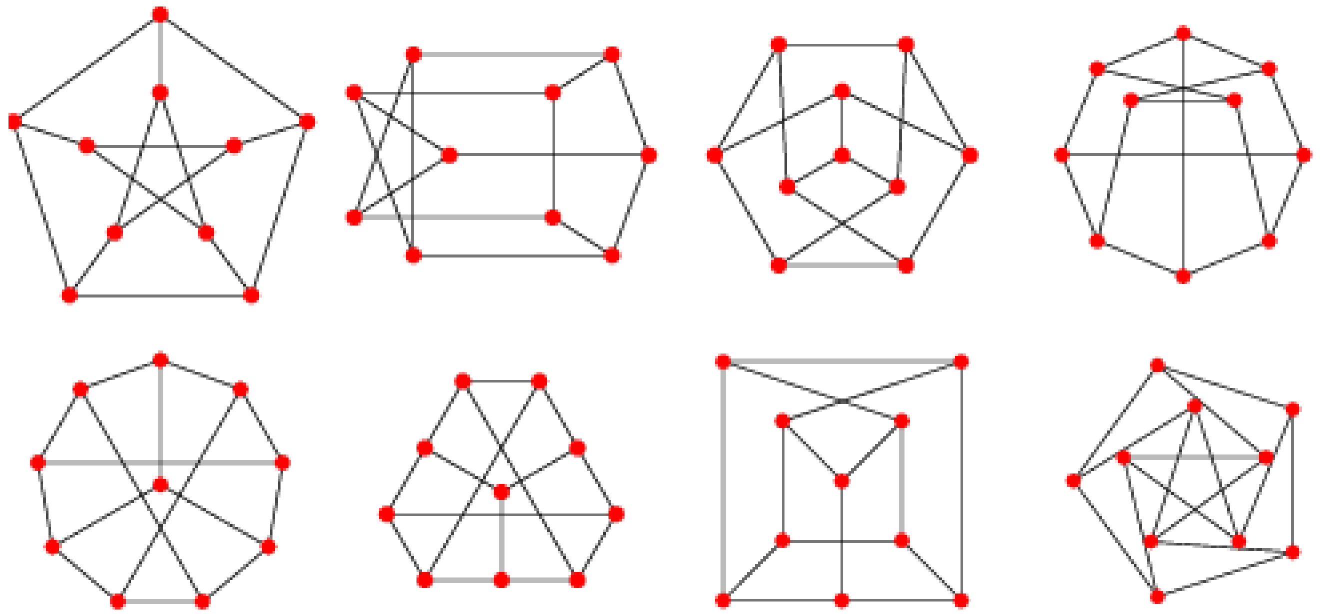
  lists

  are only used for representing graphs in computers.

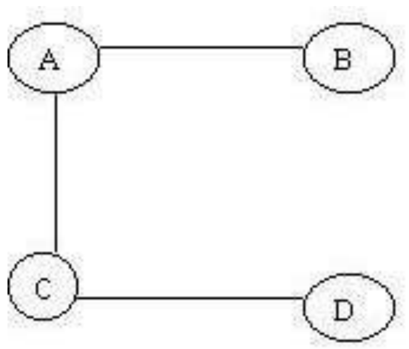
Dense Graphs and Sparse Graphs

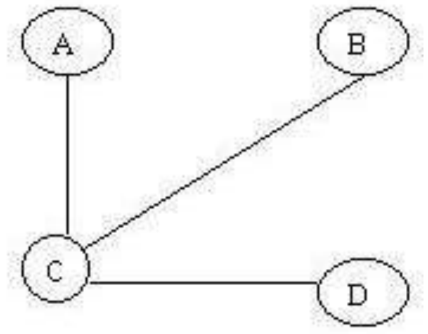
- **Dense graph** is a graph in which the number of edges is close to the maximal number of edges. The opposite, a graph with only a few edges, is a **sparse graph**.

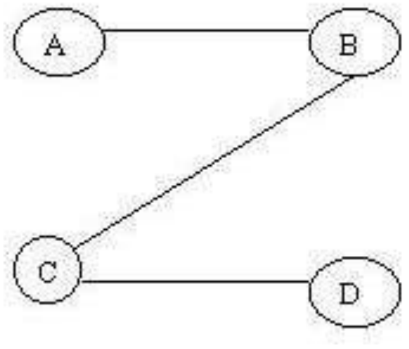
Dense Graph



Sparse Graph







Advantages of adjacency matrix

Adjacency matrix is very convenient to work with. Add (remove) an edge can be done in O(1) time, the same time is required to check, if there is an edge between two vertices.

Disadvantages of adjacency matrix

- Adjacency matrix consumes huge amount of memory for storing big graphs.
- Adjacency matrix requires huge efforts for adding/removing a vertex.
- In many algorithms you need to know the edges, adjacent to the current vertex. To draw out such an information from the adjacency matrix you have to scan over the corresponding row, which results in O(|V|) complexity.

Advantages of adjacency lists

- Adjacent list allows us to store graph in more compact form, than adjacency matrix.
- Adjacent list allows to get the list
  of adjacent vertices in O(1) time, which is a big advantage for some algorithms.

Disadvantages of adjacency lists

- Adding/removing an edge to/from adjacent list is not so easy as for adjacency matrix. It requires, on the average,O(|E| / |V|) time, which may result in cubical complexity for dense graphs to add all edges.
- If there is an edge between two vertices can be done in O(|E| / |V|) when list of adjacent vertices is unordered or $O(\log_2(|E| / |V|))$ when it is sorted. This operation stays quite cheap.

Disadvantages of adjacency lists (cont…)

- Adjacent list doesn't allow us to make an efficient implementation, if dynamically change of vertices number is required. Adding new vertex can be done in O(V), but removal results in O(E) complexity.

Conclusion

- Adjacency matrix is good for dense graphs.
- Adjacency lists is good for sparse graphs and also for changing the no of nodes.