**Graph Theory**

## ****Representing Graphs in Computer Memory****

There are two standard ways of maintaining a graph G in the memory of a computer. Two main ways are:

1. Sequential representation of G is by means of its adjacency matrix A.

Matrices are usually used when the graph G is **dense**.

2. Linked representation or adjacency structure of G, uses linked lists of neighbors.

Linked lists are usually used when G is **sparse.**

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.

(A graph G with m vertices and n edges is said to be dense when m = O(n2) and sparse when m = O(n) or even O(n log n).)

|  |  |
| --- | --- |
|  |  |

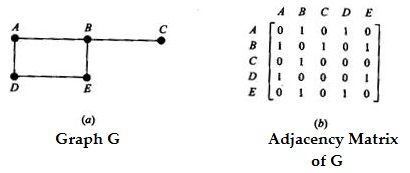
**Adjacency Matrix Representation**

The adjacency matrix of a graph G with n vertices is N x N. It is given by A = [aij].

aij = 1 if ith and jth vertices are adjacent.

= 0 if ith and jth vertices are not adjacent/otherwise.

Figure (b) contains the adjacency matrix of the graph G in Figure (a) where the vertices are ordered A, B, C, D, E. Observe that each edge {vi, vj} of G is represented twice, by aij = 1 and aji = 1. Thus, in particular, the adjacency matrix is symmetric.

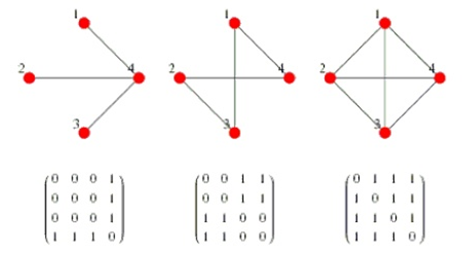


The adjacency matrix A of a graph G does depend on the ordering of the vertices of G, that is, a different ordering of the vertices yields a different adjacency matrix. However, any two such adjacency matrices are closely related in that one can be obtained from the other by simply interchanging row\* and columns. On the other hand, the adjacency matrix does not depend on the order in which the edges (pairs of vertices) are input into the computer.

There are variations of the above representation. If G is a multigraph, then we usually let denote the number of edges {vi vj}. Moreover, if G is a weighted graph, then we may let, aij denote the weight of the edge {vi, vj}.

**Example**

|  |  |
| --- | --- |
|  | Graph Adjacency matrix |

****

**Adjacency Matrix Representation**

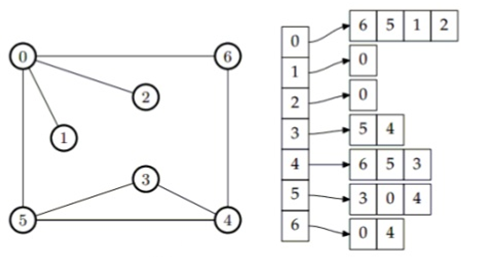
|  |  |
| --- | --- |
| The incidence matrix of a graph G with N vertices and E edges is NxE.  mij = 1 if ej is incident on vi.  = 0 otherwise |  |

 In computer’s, even there are many mathematical representations, adjacent matrix and adjacency lists are only used for representing graph in computers memory.

Example 1

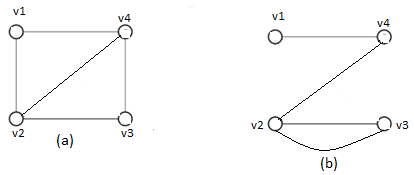
|  |  |  |
| --- | --- | --- |
| <http://d1gjlxt8vb0knt.cloudfront.net/wp-content/uploads/graph_representation12.png> | Adjacency Matrix Representation | Adjacency List Representation of Graph |
| Graph | Adjacency Matrix Representation | Adjacency List Representation in the memory. |

**Example 2**



**Example 3**

Find the adjacency matrix A = [aij] of each of the following graph G.

****

Ans.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| G1= | 0 | 1 | 0 | 1 |  | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | G2 = | 0 | 0 | 2 | 1 |
| 0 | 1 | 0 | 1 |  | 0 | 2 | 0 | 0 |
| 1 | 1 | 1 | 0 |  | 1 | 1 | 0 | 1 |
| (a) | | | | |  | (b) | | | |

Example 4

Draw the graph G corresponding to each adjacency matrix.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A= | 0 | 1 | 0 | 1 | 0 | A= | 1 | 3 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 3 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 2 | 2 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 2 | 0 |
|  | 0 | 1 | 1 | 1 | 0 |  |  |  |  |
| (a) | | | | |  | (b) | | | |

Ans.

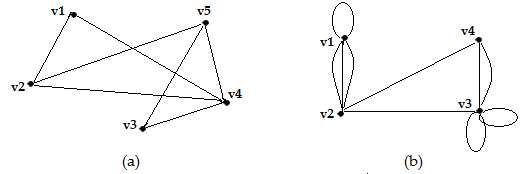
(a) Since A is a 5-square matrix, G has five vertices, say v1, v2 .. .. v5.

Draw an edge from vi to vj when aij =1. The graph appears in Fig. a.

(b) Since A is a 4-square matrix, G has four vertices, say v1, v2, v4, v4.

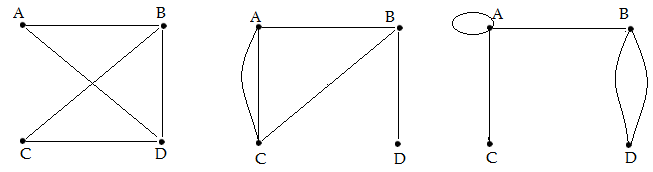
Draw n edges from vi to vj when aij = n.

Also, draw n loops at vi when aii = n. The graph appears in Fig. b.



Example 5(p228/8.63)

Find the adjacency matrix A of each graph as bellow:



Example 6(p228/8.64)

Draw the multigraph G corresponding to each of the following adjacency matrices.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A= | 0 | 2 | 0 | 1 |  | 1 | 1 | 1 | 2 |
| 2 | 1 | 1 | 1 | A = | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |  | 1 | 0 | 0 | 2 |
| 1 | 1 | 1 | 0 |  | 2 | 0 | 2 | 2 |
| (a) | | | | |  | (b) | | | |

**Linked Representation**

Let G be a graph with m vertices. The representation of G in memory by its adjacency matrix A has a number of major drawbacks.

**First** of all it may be difficult to insert or delete vertices in G. The reason is that the size of A may need to be changed and the vertices may need to be reordered, so there may be many, many changes in the matrix A.

Furthermore, suppose G is sparse. Then the matrix A will contain many zeros; hence a great deal of memory space will be wasted. Accordingly, when G is sparse, G is usually represented in memory by some type of linked representation, also called an adjacency structure.

Consider the graph G in Figure (a). Observe that G may be equivalently defined by the table in Figure (b) which shows each vertex in G followed by its adjacency list, i.e., its list of adjacent vertices (neighbors). Here the symbol Ø denotes an empty list. This table may also be presented in the compact form

G = [A:B,D;    B:A,C,D;   C:B;   D:A,B;    E: Ø]

where a colon ":" separates a vertex from its list of neighbors, and a semicolon ";" separates the different lists.

Remark: Observe that each edge of a graph G is represented twice in an adjacency structure; that is, any edge, say {A, B}, is represented by B in the adjacency list of A, and also by A in the adjacency list of B. The graph G in Figure (a) has four edges, and so there must be 8 vertices in the adjacency lists. On the other hand, each vertex in an adjacency list corresponds to a unique edge in the graph G.

|  |  |
| --- | --- |
|  |  |

The linked representation of a graph G, which maintains G in memory by using its adjacency lists, would normally contain two files (or sets of records),

One called the Vertex File, and

Other called the Edge File.