International Islamic University Chittagong
Department of Computer Science & Engineering
Spring - 2021

Course Code: CSE-2321
Course Title: Data Structures

Mohammed Shamsul Alam
Professor, Dept. of CSE, IIUC

Lecture – 16

Graphs

Graph

A graph is an abstract data structure that is used to implement the mathematical concept of graphs. It is basically a collection of vertices (also called nodes) and edges that connect these vertices.

Application of Graphs

Graphs are constructed for various types of applications such as:

- In circuit networks where points of connection are drawn as vertices and component wires become the edges of the graph.
- In transport networks where stations are drawn as vertices and routes become the edges of the graph.
- In maps that draw cities/states/regions as vertices and adjacency relations as edges.
- In program flow analysis where procedures or modules are treated as vertices and calls to these procedures are drawn as edges of the graph

Definition

A graph G is defined as an ordered set (V, E), where V(G) represents the set of vertices and E(G) represents the edges that connect these vertices.

Graph

Figure 13.1 shows a graph with V(G) = {A, B, C, D and E} and E(G) = {(A, B), (B, C), (A, D), (B, D), (D, E), (C, E)}. Note that there are five vertices or nodes and six edges in the graph

A graph can be **directed** or **undirected**. In an undirected graph, edges do not have any direction associated with them. That is, if an edge is drawn between nodes A and B, then the nodes can be traversed from A to B as well as from B to A. Figure 13.1 shows an undirected graph because it does not give any information about the direction of the edges.

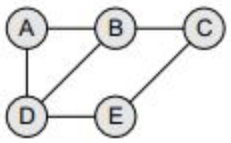

Figure 13.1 Undirected graph

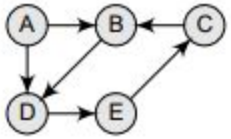

Figure 13.2 Directed graph

Look at Fig. 13.2 which shows a directed graph. In a directed graph, edges form an ordered pair. If there is an edge from A to B, then there is a path from A to B but not from B to A. The edge (A, B) is said to initiate from node A (also known as initial node) and terminate at node B (terminal node)

Graph Terminology

Adjacent nodes or neighbors:

For every edge, e = [u, v] that connects nodes u and v, the nodes u and v are the **end-points** and are said to be the **adjacent nodes** or **neighbors**.

Degree of a node:

Degree of a node u, **deg(u)**, is the total number of edges containing the node u. If **deg(u) = 0**, it means that u does not belong to any edge and such a node is known as an **isolated node**.

Path:

A path P written as P = {v_0 , v_1 , v_2 , ..., v_n), of length **n** from a node u to v is defined as a sequence of (**n+1**) nodes. Here, u = v_0 , v = v_n and v_{i-1} is adjacent to v_i for i = 1, 2, 3, ..., n.

Closed path

A path P is known as a closed path if the edge has the same end-points. That is, if $\mathbf{v_0 = v_n}$.

Simple path

A path P is known as a simple path if all the nodes in the path are distinct with an exception that v_0 may be equal to v_n . If $v_0 = v_n$, then the path is called a **closed simple path**.

Graph Terminology

Cycle:

A cycle is a closed simple path with length 3 or more. A **simple cycle** has no repeated edges or vertices (except the first and last vertices). A cycle of length k is called a **K-cycle**.

Connected graph

A graph is said to be connected if for any two vertices (u, v) in V there is a path from u to v. In a connected graph, there are no isolated nodes.

Proposition: *A graph G is connected if and only if there is a simple path between any two nodes in G*.

Complete graph

A graph G is said to be complete if every node u in G is adjacent to every other node v in G. A complete graph has **n(n–1)/2** edges, where **n** is the number of nodes in G.

Tree

A connected graph that does not have any cycle is called a tree. Therefore, a tree is treated as a special graph. If T is a finite tree with **m nodes**, then T will have **m-1 edges**. Figure 8.1(c) shows a tree.

Graph Terminology

Labelled graph or weighted graph

A graph is said to be labelled if every edge in the graph is assigned some data. In a **weighted graph**, the edges of the graph are assigned some weight or length. The weight of an edge denoted by w(e) is a positive value which indicates the cost of traversing the edge. Figure 8.1(d) shows a weighted graph.

Multiple edges

Distinct edges which connect the same end-points are called multiple edges. That is, e = [u, v] and e' = [u, v] are known as multiple edges of G.

Loop

An edge that has identical end-points is called a loop. That is, e = [u, u].

Multi-graph

A graph with multiple edges and/or loops is called a multi-graph. Figure 8.1(b) shows a multi-graph.

Size of a graph

The size of a graph is the total number of edges in it

Example 8.1

(a) Figure 8.1(a) is a picture of a connected graph with 5 nodes—A, B, C, D and E—and 7 edges:

$$[A, B],\quad [B, C],\quad [C, D],\quad [D, E],\quad [A, E],\quad [C, E]\quad [A, C]$$

There are two simple paths of length 2 from B to E: (B, A, E) and (B, C, E). There is only one simple path of length 2 from B to D: (B, C, D). We note that (B, A, D) is not a path, since $[A, D)$ is not an edge. There are two 4-cycles in the graph:

$$[A, B, C, E, A] \quad \text{and} \quad [A, C, D, E, A].$$

Note that $\deg(A) = 3$, since A belongs to 3 edges. Similarly, $\deg(C) = 4$ and $\deg(D) = 2$.

(b) Figure 8.1(b) is not a graph but a multigraph. The reason is that it has multiple edges—$e_4 = [B, C]$ and $e_5 = [B, C]$—and it has a loop, $e_6 = [D, D)$. The definition of a graph usually does not allow either multiple edges or loops.

(c) Figure 8.1(c) is a tree graph with $m = 6$ nodes and, consequently, $m - 1 = 5$ edges. The reader can verify that there is a unique simple path between any two nodes of the tree graph.

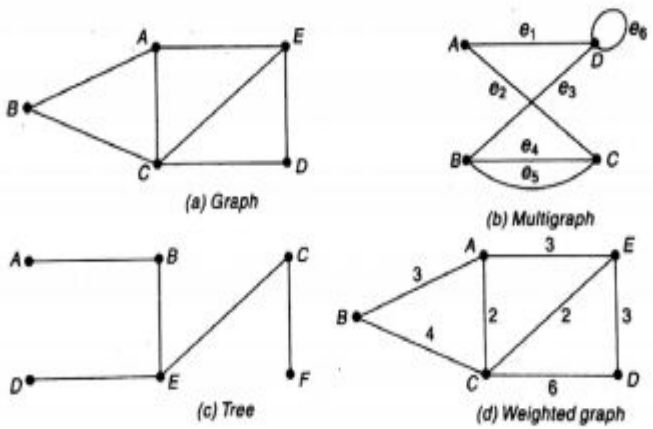

Fig. 8.1

(d) Figure 8.1(d) is the same graph as in Fig. 8.1(a), except that now the graph is weighted. Observe that $P_1 = (B, C, D)$ and $P_2 = (B, A, E, D)$ are both paths from node B to node D. Although P_2 contains more edges than P_1 the weight $w(P_2) = 9$ is less than the weight $w(P_1) = 10$.

Terminology of a Directed Graph

A directed graph G, also known as a **digraph,** is a graph in which every edge has a direction assigned to it. An edge of a directed graph is given as an ordered pair (u, v) of nodes in G.

For an edge e = (u, v) [e also called an **arc**] -

- The edge begins at u and terminates at v.
- u is known as the **origin** or **initial point** of e. Correspondingly, v is known as the **destination** or **terminal point** of e.
- u is the **predecessor** of v. Correspondingly, v is the **successor** or **neighbor** of u.
- u is adjacent to v, and v is adjacent from u.

Out-degree of a node

The out-degree of a node u, written as **outdeg(u)**, is the number of edges that originate at u.

In-degree of a node

The in-degree of a node u, written as **indeg(u)**, is the number of edges that terminate at u.

Pendant vertex (also known as leaf vertex) - A vertex with degree one.

Terminology of a Directed Graph

Source

A node u is known as a source if it has a positive out-degree but a zero in-degree.

Sink

A node u is known as a sink if it has a positive in-degree but a zero out-degree.

Reachability

A node v is said to be **reachable** from node u, if and only if there exists a (directed) path from node u to node v.

Strongly connected directed graph

A digraph is said to be strongly connected if and only if there exists a path between every pair of nodes in G. That is, if there is a path from node u to v, then there must be a path from node v to u.

Unilaterally connected graph

A digraph is said to be unilaterally connected if there exists a path between any pair of nodes u, v in G such that there is a path from u to v or a path from v to u, but not both.

Terminology of a Directed Graph

Weakly connected digraph

A directed graph is said to be weakly connected if it is connected by ignoring the direction of edges. That is, in such a graph, it is possible to reach any node from any other node by traversing edges in any direction (may not be in the direction they point). The nodes in a weakly connected directed graph must have either out-degree or in-degree of at least 1.

Parallel/Multiple edges

Distinct edges which connect the same end-points are called multiple edges. That is, e = (u, v) and e' = (u, v) are known as multiple edges of G.

Simple directed graph

A directed graph G is said to be a simple directed graph if and only if it has no parallel edges. However, a simple directed graph may contain cycles with an exception that it cannot have more than one loop at a given node.

Example 8.2

Figure 8.2 shows a directed graph G with 4 nodes and 7 (directed) edges. The edges e_2 and e_3 are said to be *parallel*, since each begins at B and ends at A. The edge e_7 is a *loop*, since it begins and ends at the same point, B. The sequence $P_1 = (D, C, B, A)$ is not a path, since (C, B) is not an edge—that is, the direction of the edge $e_5 = (B, C)$ does not agree with the direction of the path P_1. On the other hand, $P_2 = (D, B, A)$ is a path from D to A, since (D, B) and (B, A) are edges. Thus A is reachable from D. There is no path from C to any other node, so G is not strongly connected. However, G is unilaterally connected. Note that indeg$(D) = 1$ and outdeg$(D) = 2$. Node C is a sink, since indeg$(C) = 2$ but outdeg$(C) = 0$. No node in G is a source.

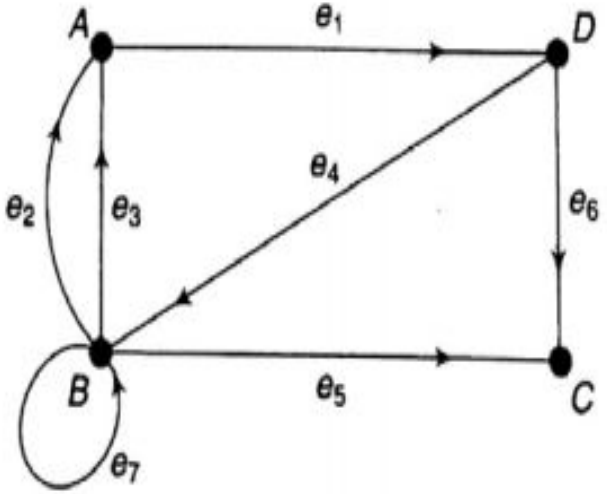

Fig. 8.2

Representation of Graphs

There are two common ways of storing graphs in the computer's memory. They are:

1. Sequential representation by using an adjacency matrix
2. Linked representation by using an adjacency list

Adjacency matrix

Suppose G is a simple directed graph with m nodes, and suppose the nodes of G have been ordered and are called $v_1, v_2, \ldots\ldots, v_m$. Then the adjacency matrix $A = (a_{ij})$ of the graph G is the m × m matrix defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } v_i \text{ is adjacent to } v_j, \text{ that is, if there is an edge } (v_i, v_j) \\ 0 & \text{otherwise} \end{cases}$$

Such a matrix A, which contains entries of only 0 and 1, is called a *bit matrix* or a *Boolean matrix*.

The adjacency matrix A of the graph G does depend on the ordering of the nodes of G; that is, a different ordering of the nodes may result in a different adjacency matrix. However, the matrices resulting from two different orderings are closely related in that one can be obtained from the other by simply interchanging rows and columns. Unless otherwise stated, we will assume that the nodes of our graph G have a fixed ordering.

Representation of Graphs

Example 8.3

Consider the graph G in Fig. 8.3. Suppose the nodes are stored in memory in a linear array DATA as follows:

DATA: X, Y, Z, W

Then we assume that the ordering of the nodes in G is as follows: $v_1 = X$, $v_2 = Y$, $v_3 = Z$ and $v_4 = W$. The adjacency matrix A of G is as follows:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Note that the number of 1's in A is equal to the number of edges in G.

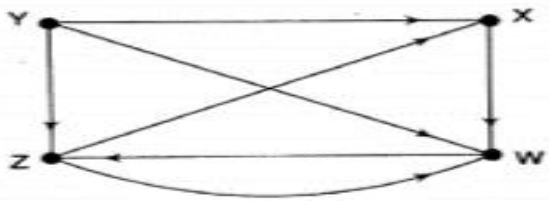

Fig. 8.3

- For a simple graph (that has no loops), the adjacency matrix has 0s on the diagonal.
- The adjacency matrix of an undirected graph is symmetric.
- The memory use of an adjacency matrix is O(n2), where n is the number of nodes in the graph.
- Number of 1s (or non-zero entries) in an adjacency matrix is equal to the number of edges in the graph.
- The adjacency matrix for a weighted graph contains the weights of the edges connecting the nodes.

Representation of Graphs

Proposition 8.2

Let A be the adjacency matrix of a graph G. Then $a_K(i, j)$, the ij entry in the matrix A^K, gives the number of paths of length K from v_i to v_j.

Consider again the graph G in Fig. 8.3, whose adjacency matrix A is given in Example 8.3. The powers A^2, A^3 and A^4 of the matrix A follow:

$$A^2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \qquad A^3 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 2 & 2 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad A^4 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 2 & 0 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Accordingly, in particular, there is a path of length 2 from v_4 to v_1, there are two paths of length 3 from v_2 to v_3, and there are three paths of length 4 from v_2 to v_4. (Here, $v_1 = X$, $v_2 = Y$, $v_3 = Z$ and $v_4 = W$.)

Suppose we now define the matrix B_r as follows:

$$B_r = A + A^2 + A^3 + \cdots + A^r$$

Then the ij entry of the matrix B_r gives the number of paths of length r or less from node v_i to v_j.

Representation of Graphs

Path Matrix

Let G be a simple directed graph with m nodes, $v_1, v_2, \ldots, v_m$. The *path matrix* or *reachability matrix* of G is the m-square matrix $P = (p_{ij})$ defined as follows:

$$P_{ij} = \begin{cases} 1 & \text{if there is a path from } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Suppose there is a path from v_i to v_j. Then there must be a simple path from v_i to v_j when $v_i \neq v_j$, or there must be a cycle from v_i to v_j when $v_i = v_j$. Since G has only m nodes, such a simple path must have length $m - 1$ or less, or such a cycle must have length m or less. This means that there is a nonzero ij entry in the matrix B_m, defined at the end of the preceding subsection. Accordingly, we have the following relationship between the path matrix P and the adjacency matrix A.

Proposition 8.3

Let A be the adjacency matrix and let $P = (p_{ij})$ be the path matrix of a digraph G. Then $P_{ij} = 1$ if and only if there is a nonzero number in the ij entry of the matrix

$$B_m = A + A^2 + A^3 + \cdots + A^m$$

Consider the graph G with $m = 4$ nodes in Fig. 8.3. Adding the matrices A, A^2, A^3 and A^4, we obtain the following matrix B_4, and, replacing the nonzero entries in B_4 by 1, we obtain the path matrix P of the graph G:

$$B_4 = \begin{pmatrix} 1 & 0 & 2 & 3 \\ 5 & 0 & 6 & 8 \\ 3 & 0 & 3 & 5 \\ 2 & 0 & 3 & 3 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

Examining the matrix P, we see that the node v_2 is not reachable from any of the other nodes.

Recall that a directed graph G is said to be *strongly connected* if, for any pair of nodes u and v in G, there are both a path from u to v and a path from v to u. Accordingly, G is strongly connected if and only if the path matrix P of G has no zero entries. Thus the graph G in Fig. 8.3 is not strongly connected.

WARSHALL'S ALGORITHM

If a graph G is given as G=(V, E), where V is the set of vertices and E is the set of edges, the path matrix of G can be found as, $P = A + A^2 + A^3 + ... + A^n$. This is a lengthy process. Warshall has given a very efficient algorithm to calculate the path matrix. Warshall's algorithm defines matrices P0 , P1 , P2 , … , Pn as given in Fig.

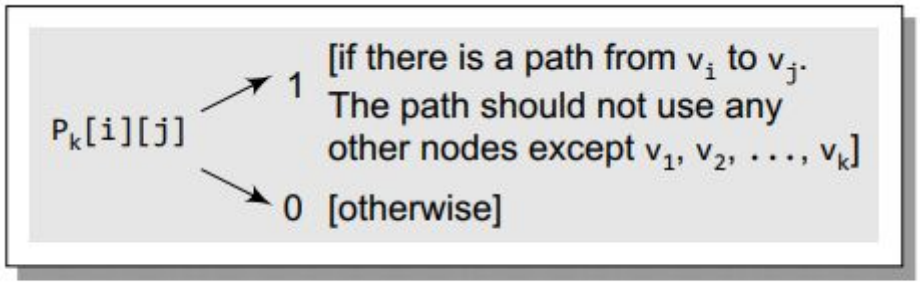

Figure 13.37 Path matrix entry

P_k [i][j] is equal to 1 only when either of the two following cases occur:

1. There is a path from vi to vj that does not use any other node except v1 , v2 , ..., vk–1. Therefore, P_{k-1}[i][j] = 1.

2. There is a path from vi to vk and a path from vk to vj where all the nodes use v1 , v2 , ..., vk–1.

Therefore, P_{k-1}[i][k] = 1 AND P_{k-1}[k][j] = 1

WARSHALL'S ALGORITHM

Hence, the path matrix Pn can be calculated with the formula given as: $P_k[i][j] = P_{k-1}[i][j] \vee (P_{k-1}[i][k] \wedge P_{k-1}[k][j])$
where V indicates logical OR operation and Λ indicates logical AND operation.

Algorithm 8.1: (Warshall's Algorithm) A directed graph G with M nodes is maintained in memory by its adjacency matrix A. This algorithm finds the (Boolean) path matrix P of the graph G.

1. Repeat for I, J = 1, 2,..., M: [Initializes P.]
   If A[I, J] = 0, then: Set P[I, J] := 0;
   Else: Set P[I, J] := 1.
   [End of loop.]
2. Repeat Steps 3 and 4 for K = 1, 2, ..., M: [Updates P.]
3. Repeat Step 4 for I = 1, 2, ..., M:
4. Repeat for J = 1, 2, ..., M:
   Set P[I, J] : = P[I, J] ∨ (P[I, K] ∧ P[K, J]).
   [End of loop.]
   [End of Step 3 loop.]
   [End of Step 2 loop.]
5. Exit.

WARSHALL'S ALGORITHM

8.8 Consider the graph G in Fig. 8.21 and its adjacency matrix A obtained in Problem 8.7. Find the path matrix P of G using Warshall's algorithm rather than the powers of A.

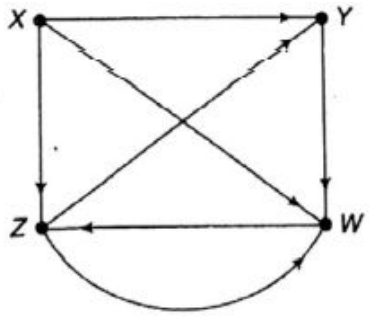

Fig. 8.21

Then:

$$P_1 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad P_2 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$P_3 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad P_4 = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Observe that $P_0 = P_1 = P_2 = A$. The changes in P_3 occur for the following reasons:

$P_3(4, 2) = 1$	because	$P_2(4, 3) = 1$	and	$P_2(3, 2) = 1$
$P_3(4, 4) = 1$	because	$P_2(4, 3) = 1$	and	$P_2(3, 4) = 1$

The changes in P_4 occur similarly. The last matrix, P_4, is the required path matrix P of the graph G.