Graph Theory

CHAPTER.10

PRESENTED BY

F. SULEIMAN – H. WOROOD – A. TAGREED

SUPERVISED

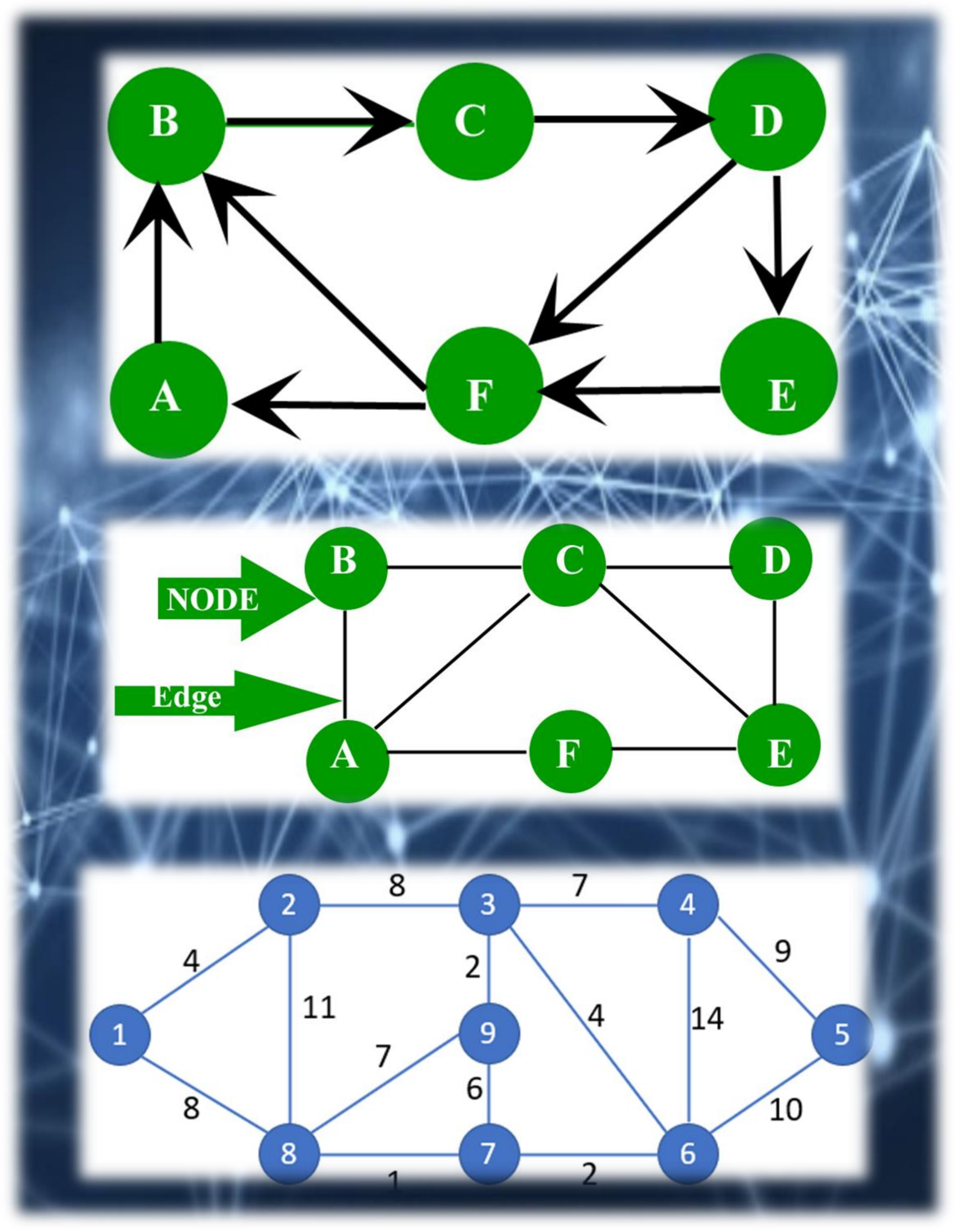
DR. IBRAHIM NADHEER

Outlines

- INTRODUCTION
- 10.2 DEFINITIONS
  - GRAPH
  - WEIGHTED GRAPH
  - GRAPH REPRESENTATION
- 10.3 COMPUTING THE CONNECTIVITY MATRIX
- 10.4 FINDING CONNECTED COMPONENTS
- 10.5 ALL-PAIRS SHORTEST PATHS

Introduction

operations research, where graphs play a crucial role in modeling and solving numerous optimization problems such as scheduling, routing, transportation, and network flow problems. It is therefore important for these applications to develop efficient algorithms to manipulate graphs and answer questions about them. As a consequence, a large body of literature exists today on computational graph-theoretic problems and their solutions.



GRAPH

A graph [G] consists of a finite set of nodes [Vertices: V] and a finite set of [Edges : E] connecting pairs of these nodes.

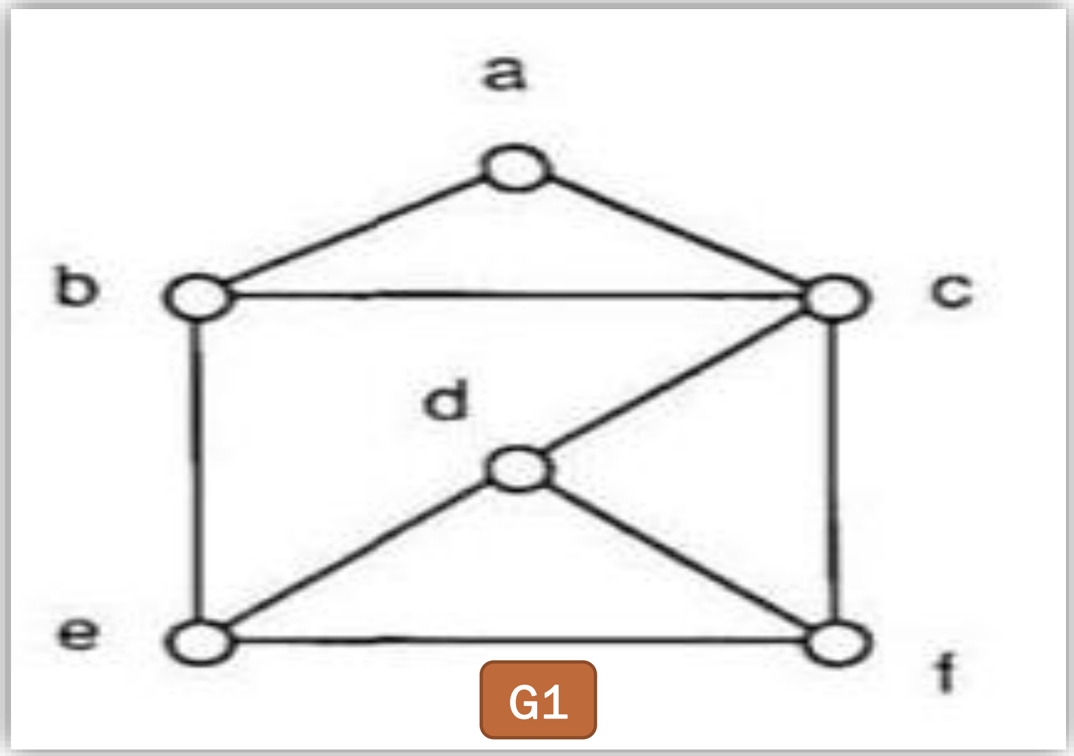
$$G = (V, E);\ V = \{v_0, v_1, \ldots, v_{n-1}\}$$

Undirected Graph

A graph G1 is undirected if E is a set of edges:

between node (a) and node (b) as **(a,b)** edge; between node (b) and node (c) as **(b,c)** edge; between node (b) and node (e) as **(b,e)** edge; and so on.

edge (u, v) = (v, u); for all v, (v, v) $\notin$ E , i.e. No self loops.



Directed Graph

A graph G2 is directed --> E is oriented and one-way connection [arrow heads]

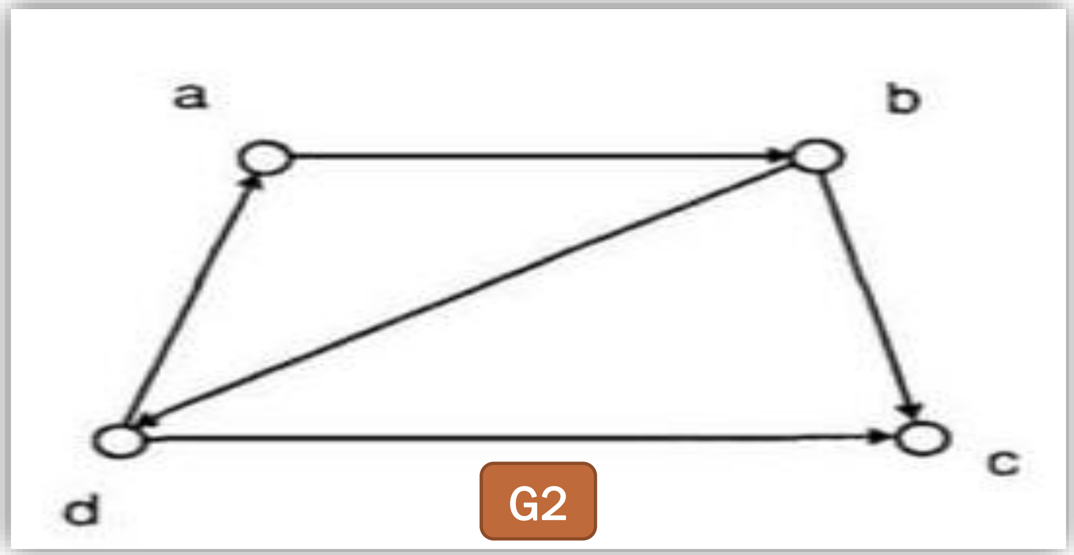
Directed (u, v) is edge from *u* to *v*, denoted as $u \rightarrow v$ *and* Self loops are allowed.

Node (a) is connected to (b)

Node (b) is connected to (c) and (d)

Node (d) is connected to (c)

But (c) is not connected to any node



Weighted Graph

Weighted Undirected Graph

A graph G1 is undirected if E is a set of edges:

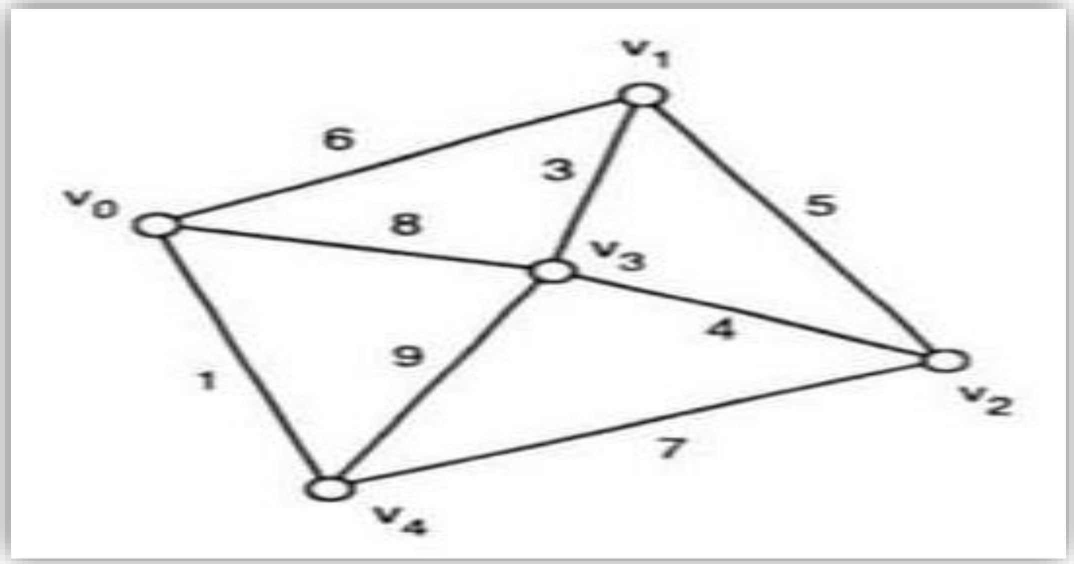
between node (V_0) and node (V_1) as **(V_1,V_2)** edge = **6,**

between node (V_0) and node (V_4) as **(V_1,V_4)** edge = **1,** and so on.

edge (u, v) = (v, u); for all v, (v, v) $\notin$ E , i.e. No self loops.

each edge has an associated **weight**, given by a weight function

$$w : E \rightarrow \text{R, R is real number}$$



G1

Weighted Directed Graph

A graph G2 is directed --> E is oriented and one-way connection [**arrow heads**]

Directed (u, v) is edge from *u* to *v*, denoted as $u \rightarrow v$ *and* **Self loops are allowed.**

Node (V_0) is connected to (V_4) with weight = **7,**

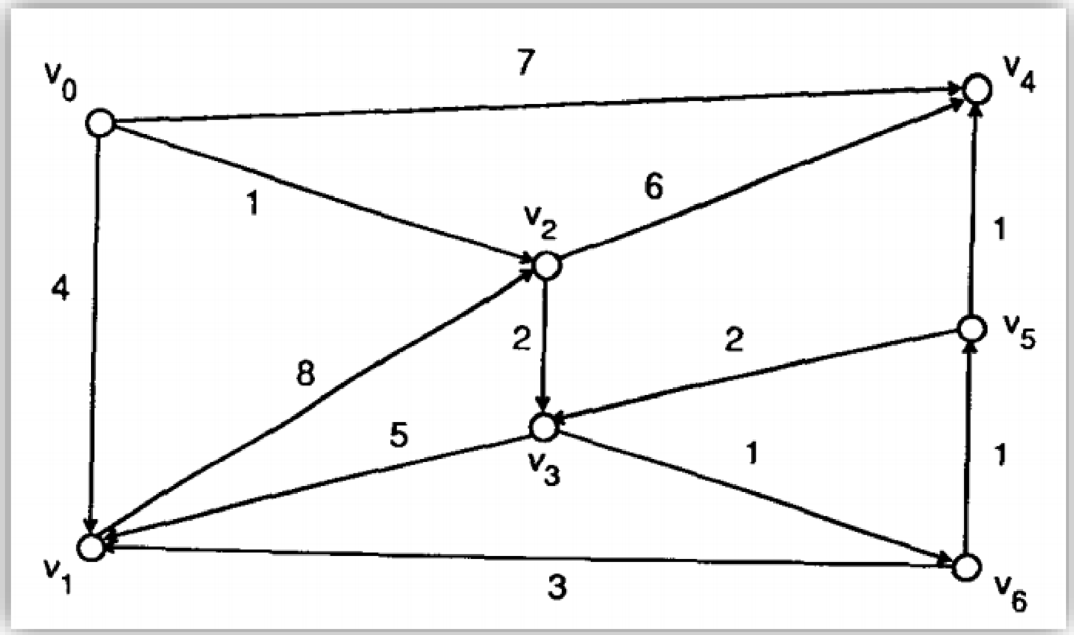
Node (V_1) is connected to (V_2) with weight = **8,**

Node (V_6) is connected to (V_5) with weight = **1, and so on.**

But (V_4) is not connected to any node

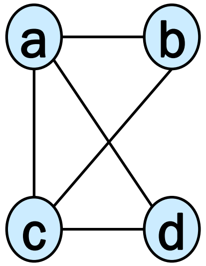
each edge has an associated **weight**, given by a weight function

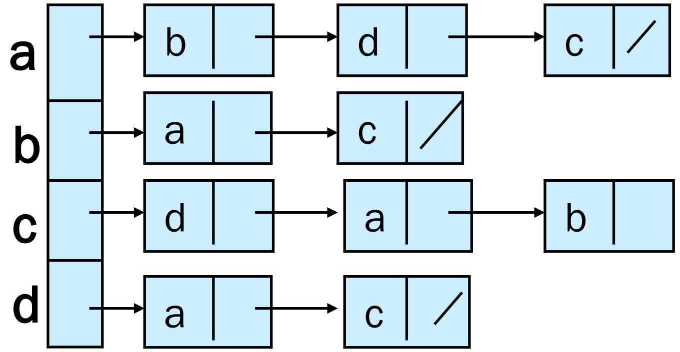
$$w : E \rightarrow \text{R, R is real number}$$



G2

Graph Representation





Adjacency Lists

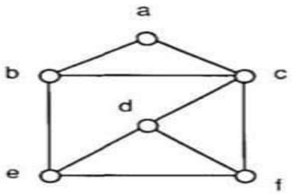
a → (a,b) – (a,d) – (a,c)
b → (b,a) – (b,c)
c → (c,d) – (c,a) – (c,b)
d → (d,a) – (d,c)

Adjacency MATRIX

$$a_{i,j} = \begin{cases} 1 & \text{if } v_i \textit{ is connected to } vj \\ 0 & \text{otherwise} \end{cases}$$

Undirected

Symmetric Matrix

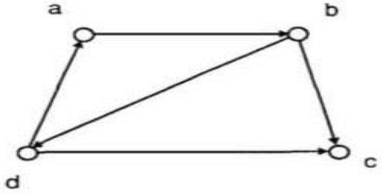


$$\begin{array}{cc|c} & & \begin{array}{cccccc} a & b & c & d & e & f \\ 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array} & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

(a) (b)

Figure 10.1 Graph with six nodes and its adjacency matrix.

Directed



$$\begin{array}{cc|c} & & \begin{array}{cccc} a & b & c & d \\ 0 & 1 & 2 & 3 \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

(a) (b)

Figure 10.2 Directed graph and its adjacency matrix.

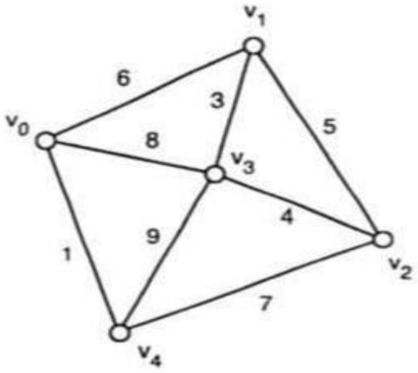
Graph Representation

Weighted MATRIX

Weight between nodes may represent distance, cost, time, probability, and so on.

Undirected

Symmetric Matrix



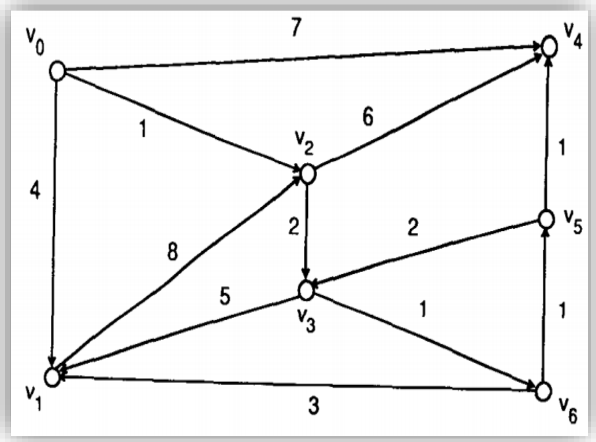
(a)

	V_0 (0)	V_1 (1)	V_2 (2)	V_3 (3)	V_4 (4)
V_0 (0)	0	6	0	8	1
V_1 (1)	6	0	5	3	0
V_2 (2)	0	5	0	4	7
V_3 (3)	8	3	4	0	9
V_4 (4)	1	0	7	9	0

(b)

Figure 10.3 Weighted graph and its weight matrix.

Directed



	V_0	V_1	V_2	V_3	V_4	V_5	V_6
V_0	0	4	1	0	7	0	0
V_1	0	0	8	0	0	0	0
V_2	0	0	0	2	6	0	0
V_3	0	5	0	0	0	0	1
V_4	0	0	0	0	0	0	0
V_5	0	0	0	2	1	0	0
V_6	0	3	0	0	0	1	0

COMPUTING CONNECTIVITY MATRIX

The connectivity matrix of an n-node graph G is an n x n matrix C whose elements are defined as follows:

$$c_{jk} = \begin{cases} 1 & \text{if there is a path of length 0 } \mathit{or\ more\ from\ vj\ to\ vk} \\ 0 & \text{otherwise} \end{cases}$$

to compute C. The approach that we take uses Boolean matrix multiplication, which differs from regular matrix multiplication in that:

(i) the matrices to be multiplied as well as the product matrix are all binary, that is each of their entries is either 0 or 1;

(ii) the Boolean (or logical) and operation replaces regular multiplication, that is, 0 and 0 = 0, 0 and I = 0, 1 and 0 = 0, and 1 and 1 = 1; and

(iii) the Boolean (or logical) or operation replaces regular addition, that is, 0 or 0 = 0, 0 or 1 = 1, 1 or 0 = 1, and 1 or 1 = 1.

Thus if X, X, and Z are n x n Boolean matrices where Z is the Boolean product of X d x then

z_{ij} = (x_{i1} and y_{1j}) or (x_{i2} and Y_{2J}) or . . . or (x_{in} and y_{nj}) for i, j = 0, 1, ..., n -1

$b_{jk} = a_{jk}$ (for $j \neq k$) and $b_{ij} = 1$

$$b_{jk} = \begin{cases} 1 & \text{if there is a path of length 0 } \mathit{or\ more\ from\ vj\ to\ vk} \\ 0 & \text{otherwise} \end{cases}$$

procedure CUBE CONNECTIVITY (A, C)

Step 1: {The diagonal elements of the adjacency matrix are made equal to 1}
for $j = 0$ **to** $n - 1$ **do in parallel**
$A(0, j, j) \leftarrow 1$
end for.

Step 2: {The A registers are copied into the B registers}
for $j = 0$ **to** $n - 1$ **do in parallel**
for $k = 0$ **to** $n - 1$ **do in parallel**
$B(0, j, k) \leftarrow A(0, j, k)$
end for
end for.

Step 3: {The connectivity matrix is obtained through repeated Boolean multiplication}
for $i = 1$ **to** $\lceil \log(n - 1) \rceil$ **do**
(3.1) CUBE MATRIX MULTIPLICATION (A, B, C)
(3.2) **for** $j = 0$ **to** $n - 1$ **do in parallel**
for $k = 0$ **to** $n - 1$ **do in parallel**
(i) $A(0, j, k) \leftarrow C(0, j, k)$
(ii) $B(0, j, k) \leftarrow C(0, j, k)$
end for
end for
end for. □

Analysis: It follows that the overall running time of this procedure is:
t(n) = O(log²n), Since p(n) = n³;
c(n) = O(n³ log² n)

COMPUTING CONNECTIVITY MATRIX

Example 10.1 Consider the adjacency matrix in Fig. 10.2(b). After steps 1 and 2 of procedure CUBE CONNECTIVITY, we have computed.

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

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

The first iteration of step 3 produce:

$$B^2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

while the second yields $B^4 = B^2$ □

procedure CUBE CONNECTIVITY (A, C)

Step 1: {The diagonal elements of the adjacency matrix are made equal to 1}
for $j = 0$ **to** $n - 1$ **do in parallel**
  $A(0, j, j) \leftarrow 1$
end for.

Step 2: {The A registers are copied into the B registers}
for $j = 0$ **to** $n - 1$ **do in parallel**
  for $k = 0$ **to** $n - 1$ **do in parallel**
    $B(0, j, k) \leftarrow A(0, j, k)$
  end for
end for.

Step 3: {The connectivity matrix is obtained through repeated Boolean multiplication}
for $i = 1$ **to** $\lceil \log(n - 1) \rceil$ **do**
  (3.1) CUBE MATRIX MULTIPLICATION (A, B, C)
  (3.2) **for** $j = 0$ **to** $n - 1$ **do in parallel**
    for $k = 0$ **to** $n - 1$ **do in parallel**
      (i) $A(0, j, k) \leftarrow C(0, j, k)$
      (ii) $B(0, j, k) \leftarrow C(0, j, k)$
    end for
  end for
end for. □

Boolean Operations
X → AND
\+ → OR

Analysis: It follows that the overall running time of this procedure is:
$t(n) = O(\log^2 n)$, Since $p(n) = n^3$;
$c(n) = O(n^3 \log^2 n)$

FINDING CONNECTED COMPONENTS

An **undirected** graph is said to be **connected** if for every pair v_i and v_j of its vertices there is a path from v_i to v_j.

An **undirected** graph is said to be **Fully-connected** if:

1. *Every pair in graph vi and vj of its vertices there is a path from vi to vj.*
2. *Every edge in graph (vi,vj) of its edges, there is a edge (vj, vi).*

A connected component of a graph G is a subgraph G' of G that is connected. The problem we consider in this section is the following. An undirected n-node graph G is given by its adjacency matrix, and it is required to decompose G into the smallest possible number of connected components. We can solve the problem by first computing the connectivity matrix C of G. Using C,

we can now construct an n x n matrix D whose entries are defined by:

$$d_{jk} = \begin{cases} V_k & \text{if } c_{jk} = 1 \\ 0 & \text{otherwise} \end{cases}$$

procedure CUBE COMPONENTS (A, C)

Step 1: {Compute the connectivity matrix}
CUBE CONNECTIVITY (A, C).

Step 2: {Construct the matrix D}
for $j = 0$ **to** $n-1$ **do in parallel**
  for $k = 0$ **to** $n-1$ **do in parallel**
    if $C(0, j, k) = 1$ **then** $C(0, j, k) = v_k$
    end if
  end for
end for.

Step 3: {Assign a component number to each vertex}
for $j = 0$ **to** $n-1$ **do in parallel**
  (3.1) the n processors in row j (forming a log n-dimensional cube) find the smallest l for which $C(0, j, l) \neq 0$
  (3.2) $C(0, j, 0) \leftarrow l$
end for. □

Analysis: As shown, step1 requires $O(\log^2 n)$ time, steps 2 and 3.2 take constant time. Step 3.1 can be done in $O(\log n)$ time. The overall running time of procedure CUBE COMPONENTS:
$t(n) = O(\log^2 n)$, since $p(n) = n^3$
$c(n) = O(n^3 \log^2 n)$.

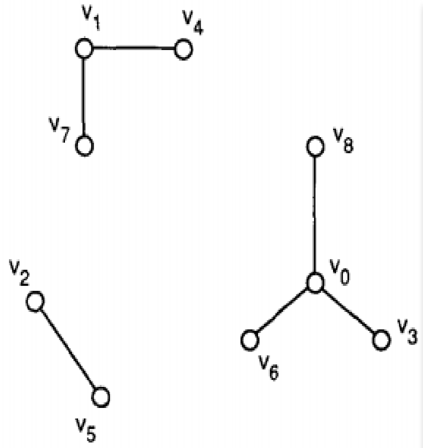
FINDING CONNECTED COMPONENTS

Example 10.2 Consider the graph in Fig. 10.5(a) whose adjacency and connectivity matrices are given Figs. 10.5(b) and (c), respectively. Matrix D is shown in Fig. 10.5(d). The component assignment is therefore:

component 0: V_0, V_3, V_6, V_8

component 1: V_1, V_4, V_7

component 2: V_2, V_5



(a)

		V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
		0	1	2	3	4	5	6	7	8
V_0	0	0	0	0	1	0	0	1	0	1
V_1	1	0	0	0	0	1	0	0	1	0
V_2	2	0	0	0	0	0	1	0	0	0
V_3	3	1	0	0	0	0	0	0	0	0
V_4	4	0	1	0	0	0	0	0	0	0
V_5	5	0	0	1	0	0	0	0	0	0
V_6	6	1	0	0	0	0	0	0	0	0
V_7	7	0	1	0	0	0	0	0	0	0
V_8	8	1	0	0	0	0	0	0	0	0

		V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
		0	1	2	3	4	5	6	7	8
V_0	0	1	0	0	1	0	0	1	0	1
V_1	1	0	1	0	0	1	0	0	1	0
V_2	2	0	0	1	0	0	1	0	0	0
V_3	3	1	0	0	1	0	0	1	0	1
V_4	4	0	1	0	0	1	0	0	1	0
V_5	5	0	0	1	0	0	1	0	0	0
V_6	6	1	0	0	1	0	0	1	0	1
V_7	7	0	1	0	0	1	0	0	1	0
V_8	8	1	0	0	1	0	0	1	0	1

(c)

	V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
	0	1	2	3	4	5	6	7	8
0	v_0	0	0	v_3	0	0	v_6	0	v_8
1	0	v_1	0	0	v_4	0	0	v_7	0
2	0	0	v_2	0	0	v_5	0	0	0
3	v_0	0	0	v_3	0	0	v_6	0	v_8
4	0	v_1	0	0	v_4	0	0	v_7	0
5	0	0	v_2	0	0	v_5	0	0	0
6	v_0	0	0	v_3	0	0	v_6	0	v_8
7	0	v_1	0	0	v_4	0	0	v_7	0
8	v_0	0	0	v_3	0	0	v_6	0	v_8

(d)

Figure 10.5 Computing connected components of graph.

ALL-PAIRS SHORTEST PATHS

A **directed** and **weighted** graph G = (V , E)

For every pair of vertices v_i and v_j in V it is required to find the shortest path from v_i to v_j along edges in E.

ex: the **shortest path** from v_0 to v_4

An n-vertex graph G is given by its n x n weight matrix W, construct an n x n matrix D such that d_{1j} is the length of the shortest path from v_i to v_j in G

W has **positive, zero, or negative** entries as long as there is no cycle of negative length in G.

we can use a special form of matrix multiplication in which the standard operations of matrix multiplication, that is, x and + are replaced by + and min,

procedure CUBE SHORTEST PATHS (A, C)

Step 1: {Construct the matrix D^1 and store it in registers A and B}
- **for** $j = 0$ **to** $n - 1$ **do in parallel**
  - **for** $k = 0$ **to** $n - 1$ **do in parallel**
    - (1.1) **if** $j \neq k$ **and** $A(0, j, k) = 0$ **then** $A(0, j, k) \leftarrow \infty$ **end if**
    - (1.2) $B(0, j, k) \leftarrow A(0, j, k)$
  - **end for**
- **end for.**

Step 2: {Construct the matrices $D^2, D^4, \ldots, D^{n-1}$ through repeated matrix multiplication}
- **for** $i = 1$ **to** $\lceil \log(n - 1) \rceil$ **do**
  - (2.1) CUBE MATRIX MULTIPLICATION (A, B, C)
  - (2.2) **for** $j = 0$ **to** $n - 1$ **do in parallel**
    - **for** $k = 0$ **to** $n - 1$ **do in parallel**
      - (i) $A(0, j, k) \leftarrow C(0, j, k)$
      - (ii) $B(0, j, k) \leftarrow C(0, j, k)$
    - **end for**
  - **end for**
- **end for.** □

Special Operations
X → +
+ → min of +

Analysis: Steps 1 and 2.2 take constant time. There are [log(n - 1)1 iterations of step 2.1 each requiring O(log^n) time. The overall running time procedure CUBE SHORTEST PATHS is therefore t(n) = O(log² n), since p(n) = n³, c(n) = O(n³log²n).

ALL-PAIRS SHORTEST PATHS

(a)

	0	1	2	3	4	5	6
0	0	4	1	∞	7	∞	∞
1	∞	0	8	∞	∞	∞	∞
2	∞	∞	0	2	6	∞	∞
3	∞	5	∞	0	∞	∞	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	∞	∞	2	1	0	∞
6	∞	3	∞	∞	∞	1	0

(b)

	0	1	2	3	4	5	6
0	0	4	1	3	7	∞	∞
1	∞	0	8	10	14	∞	∞
2	∞	7	0	2	6	∞	3
3	∞	4	13	0	∞	2	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	7	∞	2	1	0	3
6	∞	3	11	3	2	1	0

(c)

	0	1	2	3	4	5	6
0	0	4	1	3	7	5	4
1	∞	0	8	10	14	12	11
2	∞	6	0	2	5	4	3
3	∞	4	12	0	3	2	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	6	14	2	1	0	3
6	∞	3	11	3	2	1	0

(d)

	0	1	2	3	4	5	6
0	0	4	1	3	6	5	4
1	∞	0	8	10	13	12	11
2	∞	6	0	2	5	4	3
3	∞	4	12	0	3	2	1
4	∞	∞	∞	∞	0	∞	∞
5	∞	6	14	2	1	0	3
6	∞	3	11	3	2	1	0

Figure 10.7 Computing all-pairs shortest paths for graph in Fig. 10.6.

```
procedure CUBE SHORTEST PATHS (A, C)
  Step 1: {Construct the matrix D^1 and store it in registers A and B}
          for j = 0 to n − 1 do in parallel
            for k = 0 to n − 1 do in parallel
              (1.1) if j ≠ k and A(0, j, k) = 0
                      then A(0, j, k) ← ∞
                    end if
              (1.2) B(0, j, k) ← A(0, j, k)
            end for
          end for.
  Step 2: {Construct the matrices D^2, D^4, ..., D^(n−1) through repeated matrix multiplication}
          for i = 1 to ⌈log(n − 1)⌉ do
              (2.1) CUBE MATRIX MULTIPLICATION (A, B, C)
              (2.2) for j = 0 to n − 1 do in parallel
                      for k = 0 to n − 1 do in parallel
                        (i)  A(0, j, k) ← C(0, j, k)
                        (ii) B(0, j, k) ← C(0, j, k)
                      end for
                    end for
          end for. □
```

Special Operations
X → +
+ → min of +

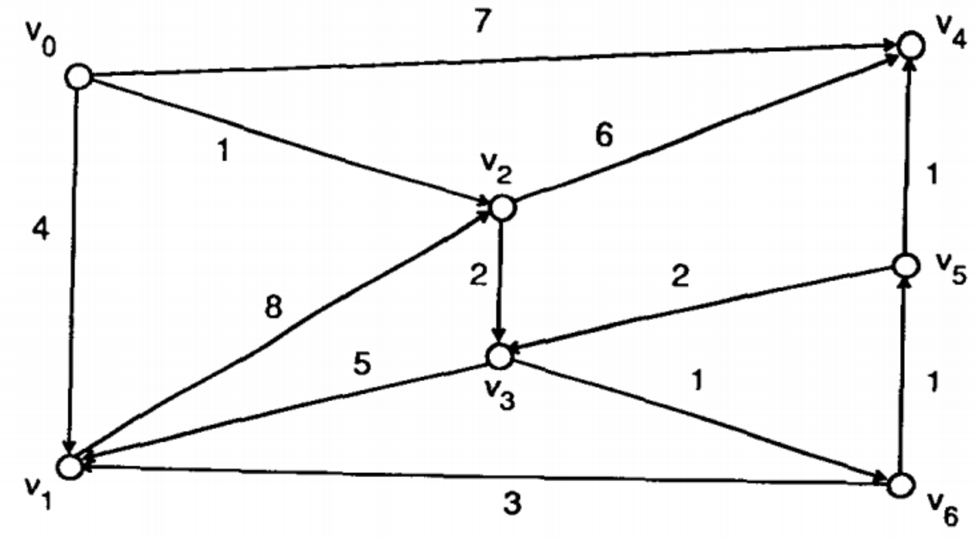


Figure 10.6: Directed and Weighted Graph

ALL-PAIRS SHORTEST PATHS

1		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	0	7	0	0
V_1	1	0	0	8	0	0	0	0
V_2	2	0	0	0	2	6	0	0
V_3	3	0	5	0	0	0	0	1
V_4	4	0	0	0	0	0	0	0
V_5	5	0	0	0	2	1	0	0
V_6	6	0	3	0	0	0	1	0

2		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

X

		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

=

3		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	3	7	∞	∞
V_1	1	∞	0	8	10	14	∞	∞
V_2	2	∞	7	0	2	6	∞	3
V_3	3	∞	4	13	0	∞	2	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	7	∞	2	1	0	3
V_6	6	∞	3	11	3	2	1	0

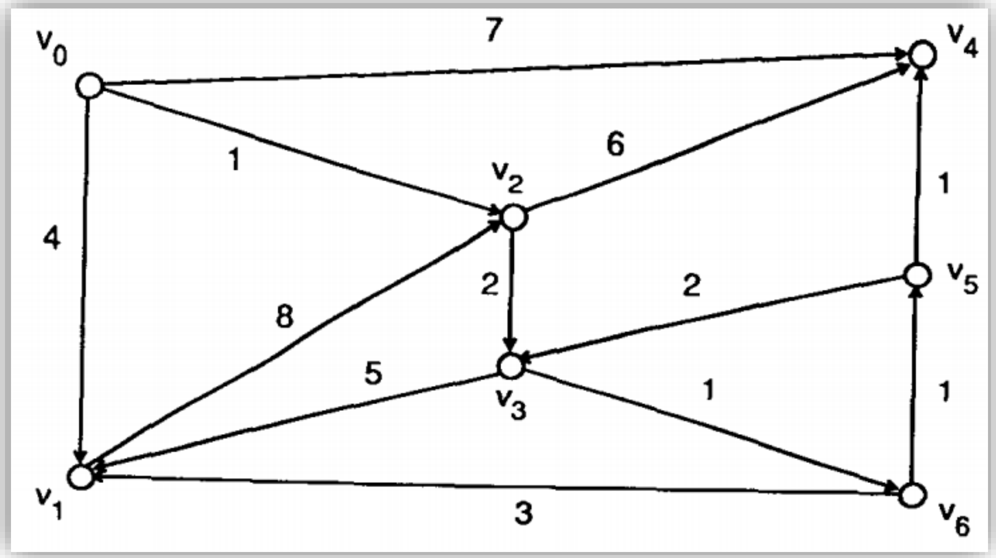


Figure 10.6: Directed and Weighted Graph

ALL-PAIRS SHORTEST PATHS

0	4	1	∞	7	∞	∞

Sum → Min

0	∞	∞	∞	∞	∞	∞

(0 + 0) , (4+∞) , (1+∞) , (∞+∞) , (7+∞) , (∞+∞) , (∞+∞)

0 , ∞ , ∞ , ∞ , ∞ , ∞ , ∞

Min of {0, ∞, ∞, ∞, ∞, ∞, ∞} = 0

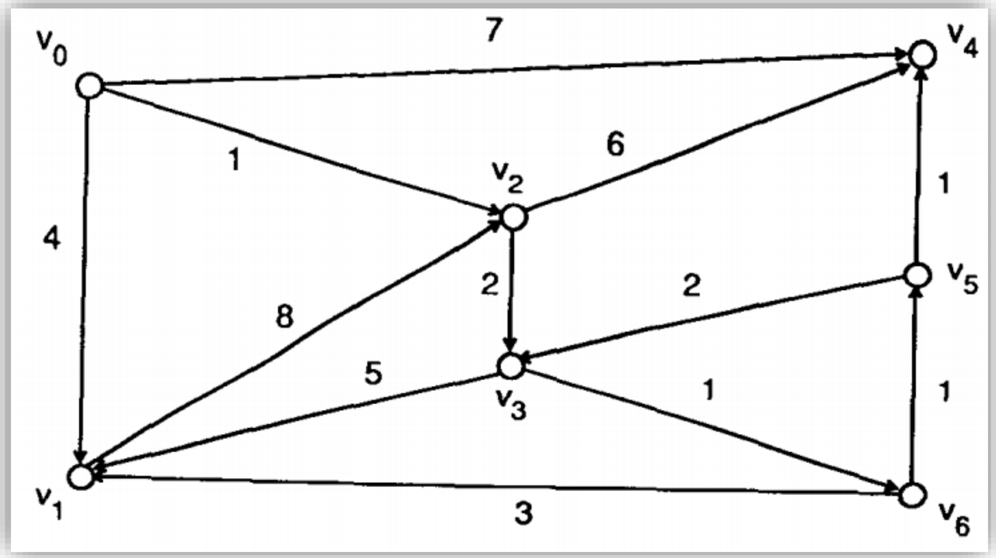


Figure 10.6: Directed and Weighted Graph

		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

X

		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

=

3		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	3	7	∞	∞
V_1	1	∞	0	8	10	14	∞	∞
V_2	2	∞	7	0	2	6	∞	3
V_3	3	∞	4	13	0	∞	2	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	7	∞	2	1	0	3
V_6	6	∞	3	11	3	2	1	0

ALL-PAIRS SHORTEST PATHS

1		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	0	7	0	0
V_1	1	0	0	8	0	0	0	0
V_2	2	0	0	0	2	6	0	0
V_3	3	0	5	0	0	0	0	1
V_4	4	0	0	0	0	0	0	0
V_5	5	0	0	0	2	1	0	0
V_6	6	0	3	0	0	0	1	0

2		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	∞	7	∞	∞
V_1	1	∞	0	8	∞	∞	∞	∞
V_2	2	∞	∞	0	2	6	∞	∞
V_3	3	∞	5	∞	0	∞	∞	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	∞	∞	2	1	0	∞
V_6	6	∞	3	∞	∞	∞	1	0

3		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	3	7	∞	∞
V_1	1	∞	0	8	10	14	∞	∞
V_2	2	∞	7	0	2	6	∞	3
V_3	3	∞	4	13	0	∞	2	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	7	∞	2	1	0	3
V_6	6	∞	3	11	3	2	1	0

4		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	3	7	5	4
V_1	1	∞	0	8	10	14	12	11
V_2	2	∞	6	0	2	5	4	3
V_3	3	∞	4	12	0	3	2	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	6	14	2	1	0	3
V_6	6	∞	3	11	3	2	1	0

Result		V_0	V_1	V_2	V_3	V_4	V_5	V_6
		0	1	2	3	4	5	6
V_0	0	0	4	1	3	6	5	4
V_1	1	∞	0	8	10	13	12	11
V_2	2	∞	6	0	2	5	4	3
V_3	3	∞	4	12	0	3	2	1
V_4	4	∞	∞	∞	∞	0	∞	∞
V_5	5	∞	6	14	2	1	0	3
V_6	6	∞	3	11	3	2	1	0

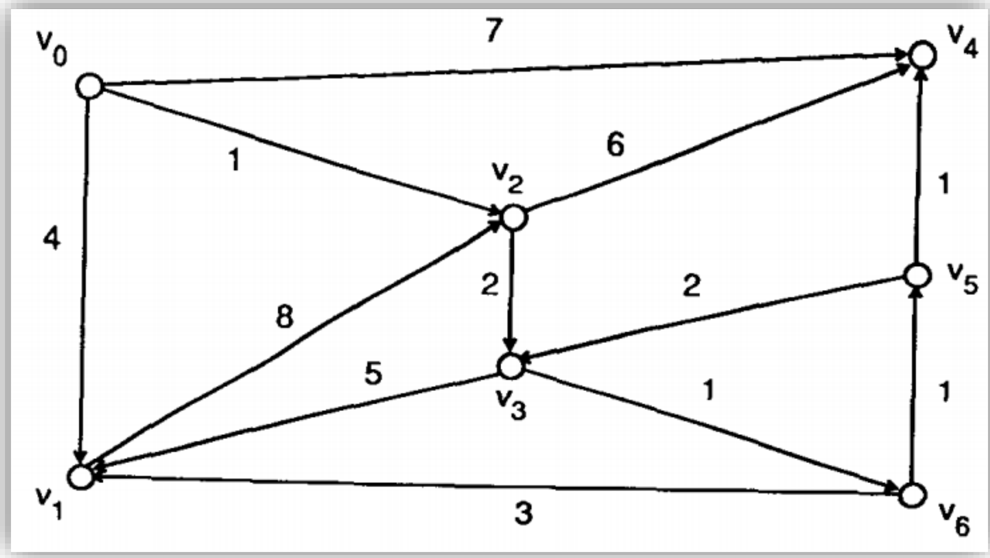


Figure 10.6: Directed and Weighted Graph

Thank YOU