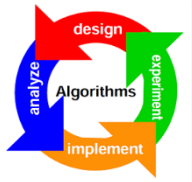
DESIGN AND ANALYSIS OF ALGORITHMS (DAA) (A34EC)

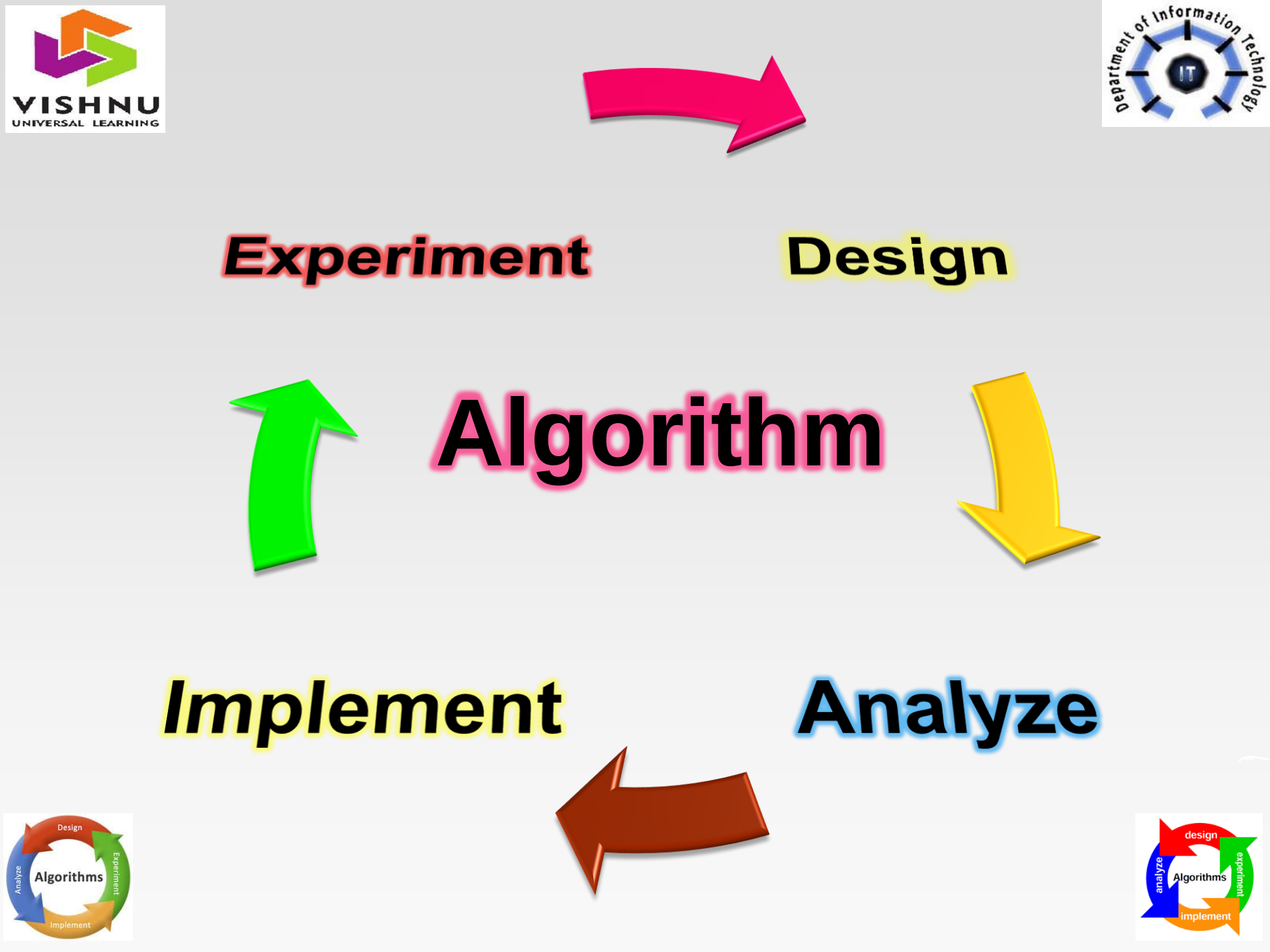
By :-

VIJAYKUMAR MANTRI,

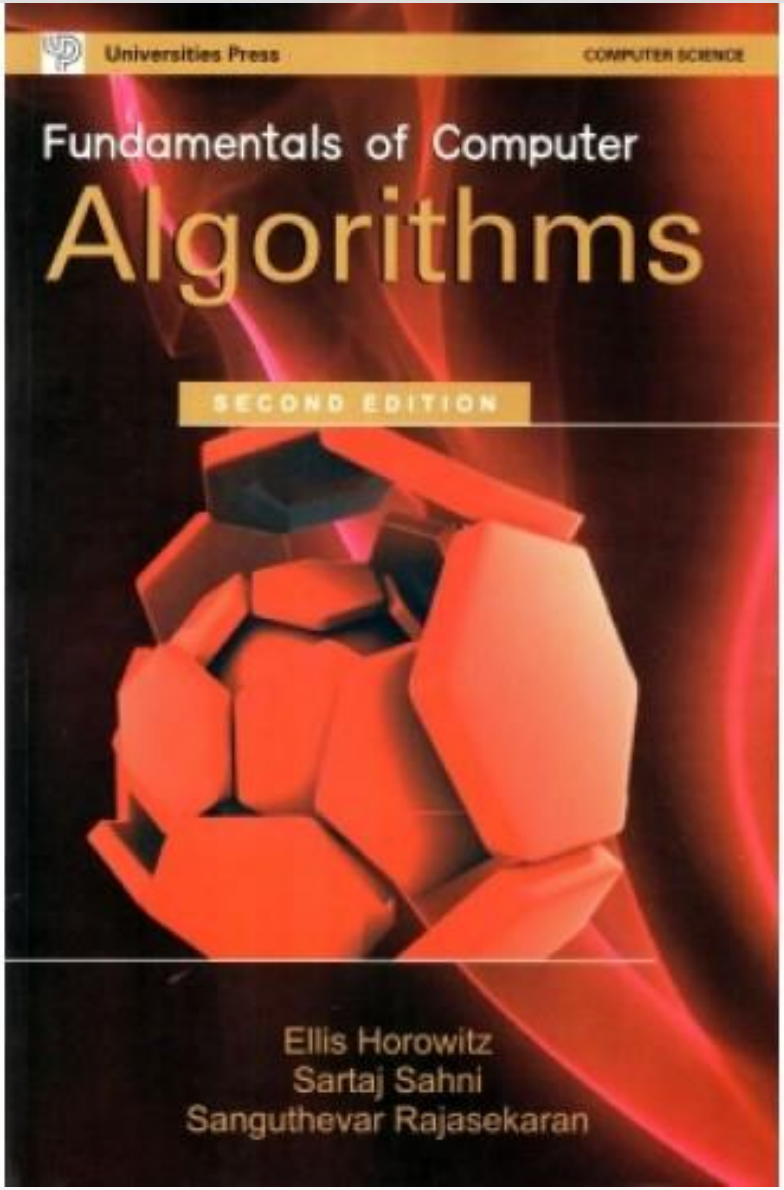
ASSOCIATE PROFESSOR.

vijay_mantri.it@bvrit.ac.in



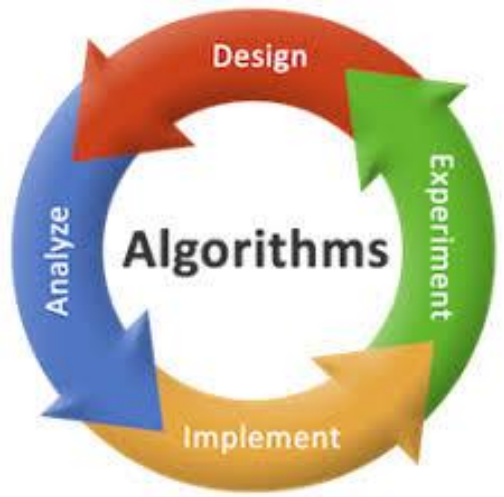

Experiment
Design
Algorithm
Implement
Analyze

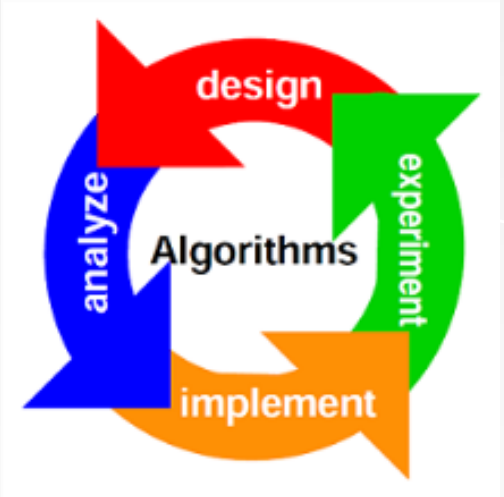
Textbook


Universities Press
COMPUTER SCIENCE
Fundamentals of Computer
Algorithms
SECOND EDITION
Ellis Horowitz
Sartaj Sahni
Sanguthevar Rajasekaran

DAA Unit IV

Dynamic Programming



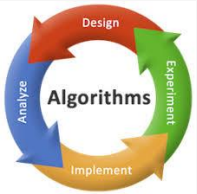


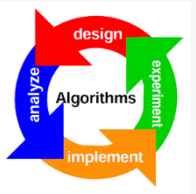
Unit IV Syllabus

Dynamic Programming:

- General Method
- Applications –

1. Multistage Graphs
2. 0/1 Knapsack Problem
3. Travelling Sales Person Problem
4. All Pairs Shortest Path Problem
5. Matrix Chain Multiplication
6. Reliability Design





Dynamic Programming–General Method

- Dynamic Programming is an algorithm design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.
- The idea of dynamic programming is quite simple: avoid calculating the same thing twice, usually by keeping a table of known result that fills up a sub instances are solved.
- In dynamic programming an optimal sequence of decisions is obtained by making explicit appeal to Principle of Optimality.
- **Principle of Optimality** :- The Principle of Optimality states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

Dynamic Programming–General Method

- Lets see differences between divide and conquer and dynamic programming methods.
- Divide and conquer is a top-down method.
- When a problem is solved by divide and conquer, we immediately attack the complete instance, which we then divide into smaller and smaller sub-instances as the algorithm progresses.
- Dynamic programming on the other hand is a bottom-up technique.
- We usually start with the smallest and hence the simplest sub-instances.
- By combining their solutions, we obtain the answers to sub-instances of increasing size, until finally we arrive at the solution of the original instances.

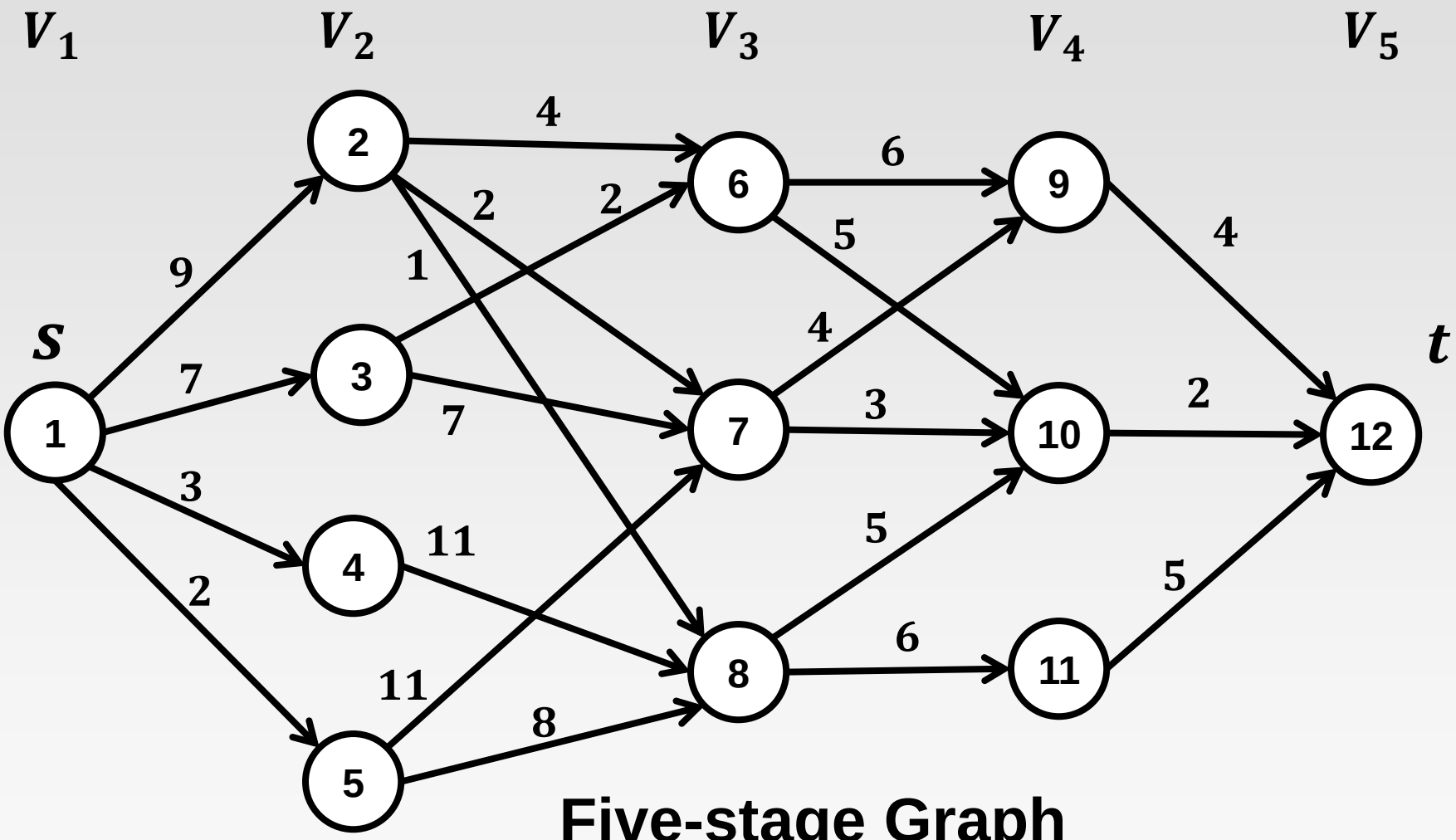
Dynamic Programming–General Method

- The essential difference between the greedy method and dynamic programming is that the greedy method only one decision sequence is ever generated.
- In dynamic programming, many decision sequences may be generated.
- However, sequences containing sub-optimal sub-sequences can not be optimal (if the principle of optimality holds) and so will not (as far as possible) be generated.

MULTISTAGE GRAPHS

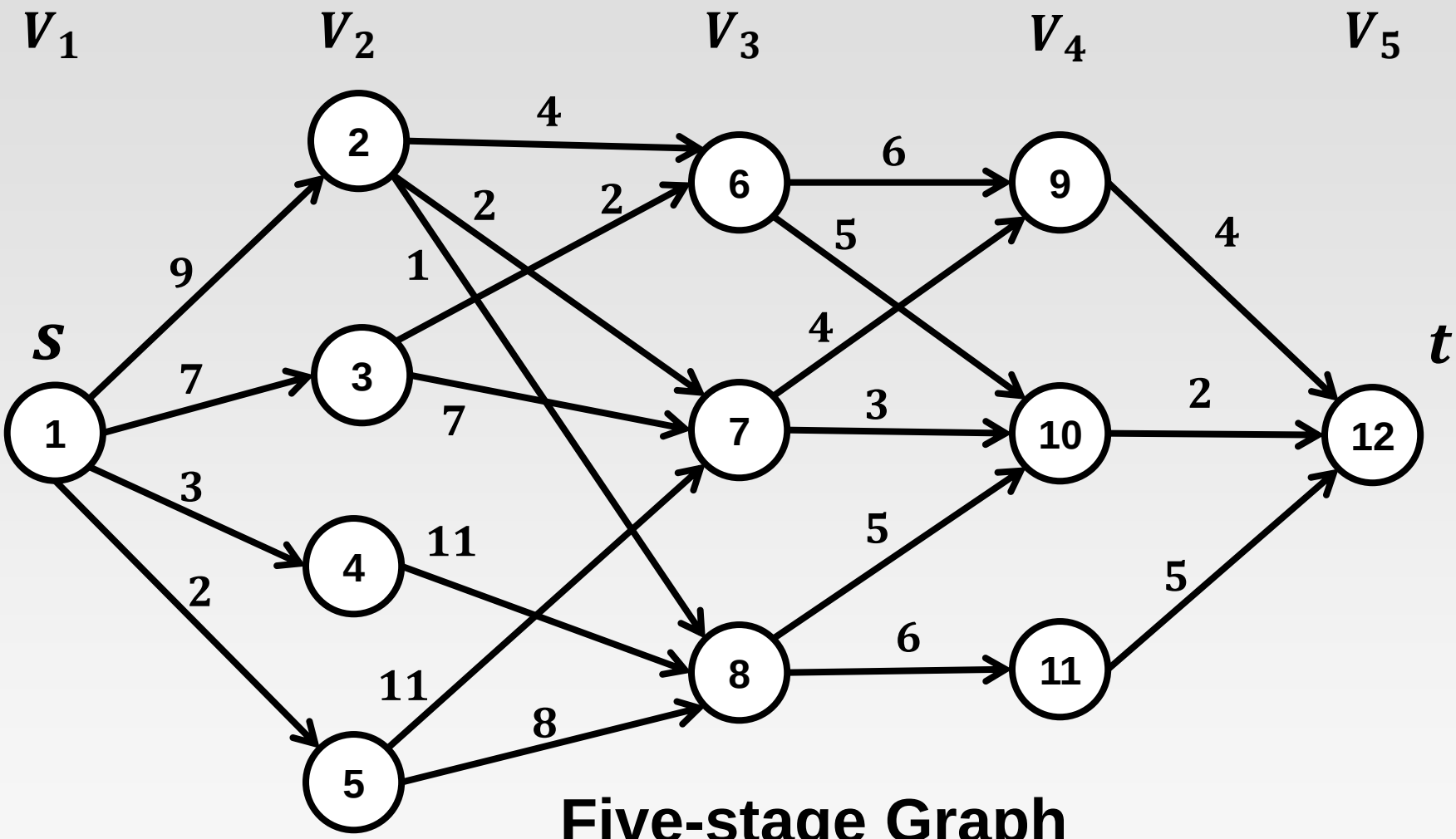
- A multistage graph $G = (V, E)$ is a directed graph in which vertices are portioned into $k \geq 2$ disjoint sets $V_i, 1 \leq i \leq k$.
- In addition, if $\langle u, v \rangle$ is an edge in E, then $u \in V_i$ and $v \in V_{i+1}$ for some $i, 1 \leq i < k$.
- The sets V_1 and V_k are such that $|V_1| = |V_k| = 1$ (i.e. number of nodes at Source & Destination is 1).
- Let 's' and 't', respectivily, be the vertices in V_1 and V_k , the vertex 's' is source and 't' the destination.
- Let $c(i, j)$ be the cost of edge $\langle i, j \rangle$.
- The cost of a path from source (s) to destination (t) is the sum of the costs of the edges on the path.
- The **Multistage Graph** problem is to find a minimum cost path from 's' to 't'.

- Each set V_i defines a stage in the graph.
- Every path from 's' to 't' starts in stage-1, goes to stage-2 then to stage-3, then to stage-4, and so on, and terminates in stage-k.



Five-stage Graph

MULTISTAGE GRAPHS



Five-stage Graph

MULTISTAGE GRAPHS

- This Mulistage Graph problem can be solved in 2 ways.
  - Forward Method.
  - Backward Method.
- A dynamic programming formulation for a $k-stage$ graph problem is obtained by first noticing that every s to t path is the result of a sequence of $k-2$ decisions.
- The ith decision involves determining which vertex in V_{i+1}, $1 \leq i \leq k-2$, is to be on the path.
- It is easy to see that the principle of optimality holds.
- Let $p(i,j)$ be a minimum-cost path from vertex j in V_i to vertex t. (i is stage and j is Vertex at stage i)
- Let $cost(i,j)$ is cost of this path from vertex j at stage i to t (Destination)

MULTISTAGE GRAPHS

- **Forward Method :-**

V	1	2	3	4	5	6	7	8	9	10	11	12
Cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

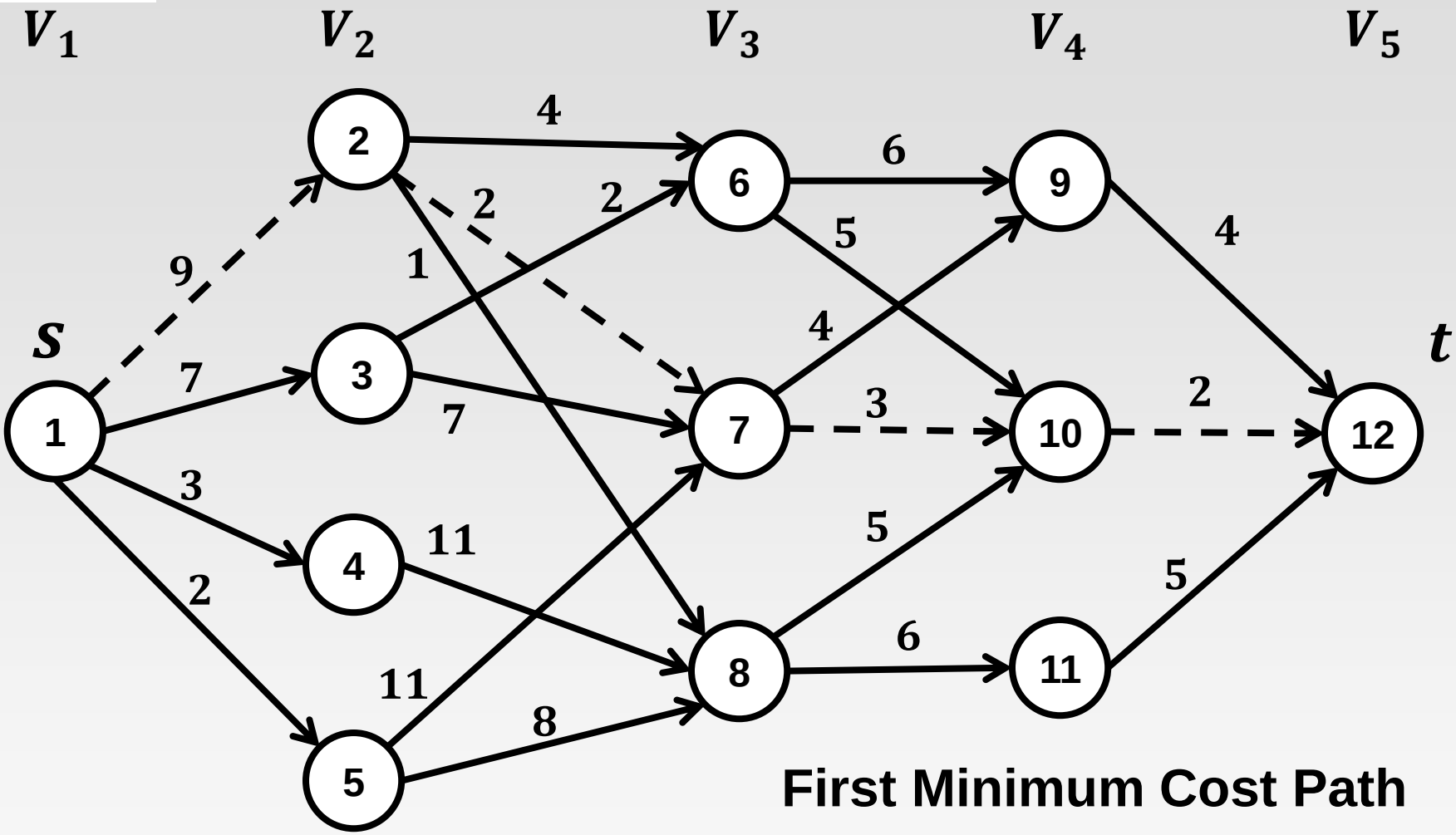
Path	1	2	7	10	12	

OR

Path	1	3	6	10	12	

- In forward method we will start from Node 12 (i.e. t), find distance from current vertex to t & come back till Node 1 (i.e. s).
- But as a Dynamic Programming for selection of Path we will use Path starting from Node 1 (i.e. s) to Node 12 (i.e. t) that's why this is called as **Forward Method.**

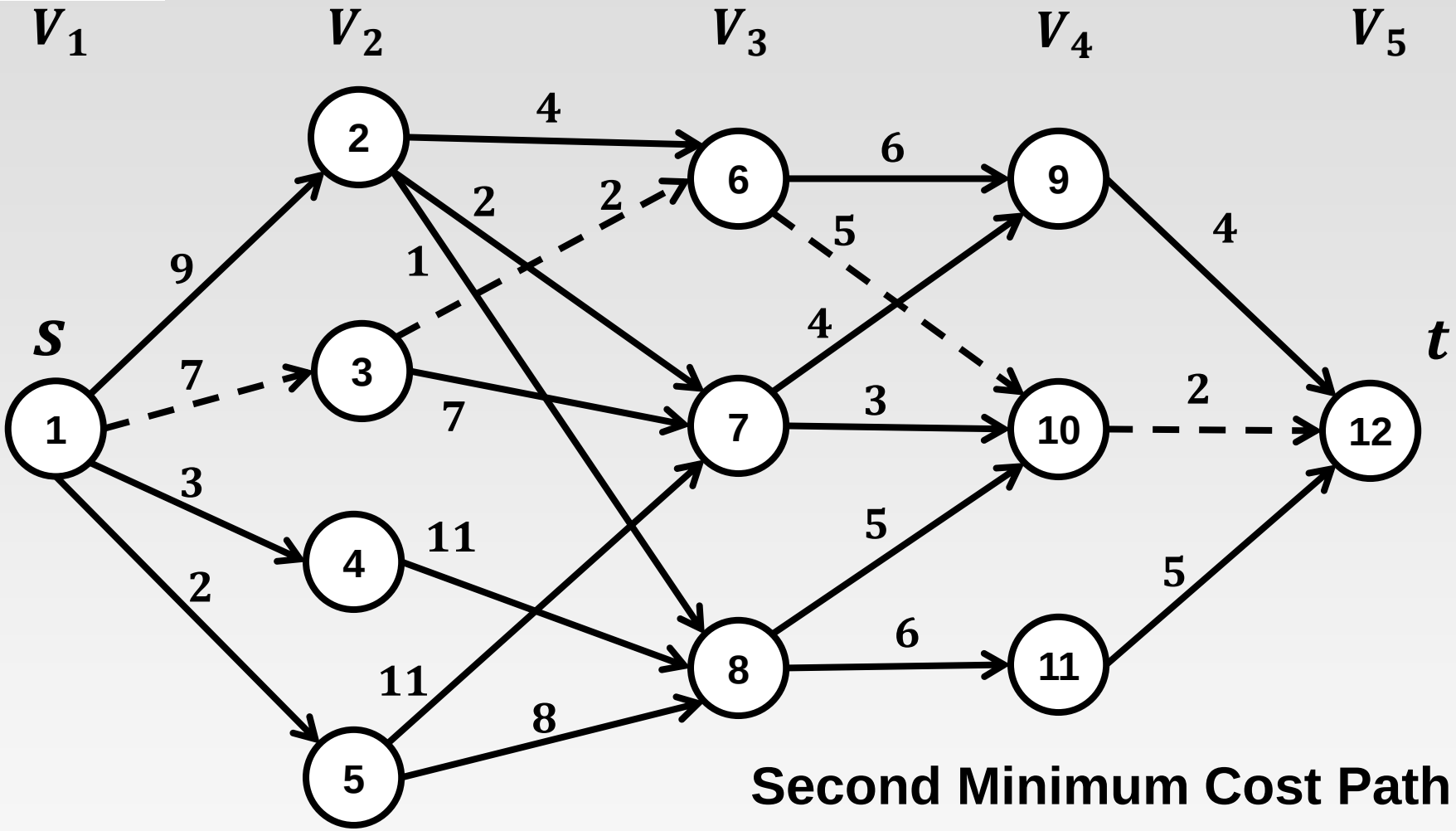
MULTISTAGE GRAPHS



First Minimum Cost Path

Five-stage Graph

MULTISTAGE GRAPHS



Second Minimum Cost Path

Five-stage Graph

- Then, using the Forward Method, we obtain

$$\boldsymbol{cost(i,j) = \min_{\substack{l \in V_{i+1} \\ \langle j,l \rangle \in E}} \{c(j,l) + cost(i+1,l)\}}$$

- Here $\boldsymbol{i}$ is Stage Number, $\boldsymbol{j}$ is vertex number. $\boldsymbol{cost(i,j)}$ is cost of path from vertex $\boldsymbol{j}$ at stage $\boldsymbol{i}$ to $\boldsymbol{t}$ (Destination)
- $\boldsymbol{c(j,l)}$ is cost of edge from $\boldsymbol{j}$ to $\boldsymbol{l}$.
- Using this formula on Graph, we obtain

$$\boldsymbol{cost(4,9) = min\{c(9,12) + cost(5,12)\}}$$
$$\boldsymbol{= min\{4 + 0\}}$$
$$\boldsymbol{= 4}$$

- Similarly,

$$\boldsymbol{cost(4,10) = min\{2 + 0\}}$$
$$\boldsymbol{= 2}$$
$$\boldsymbol{cost(4,11) = 5}$$

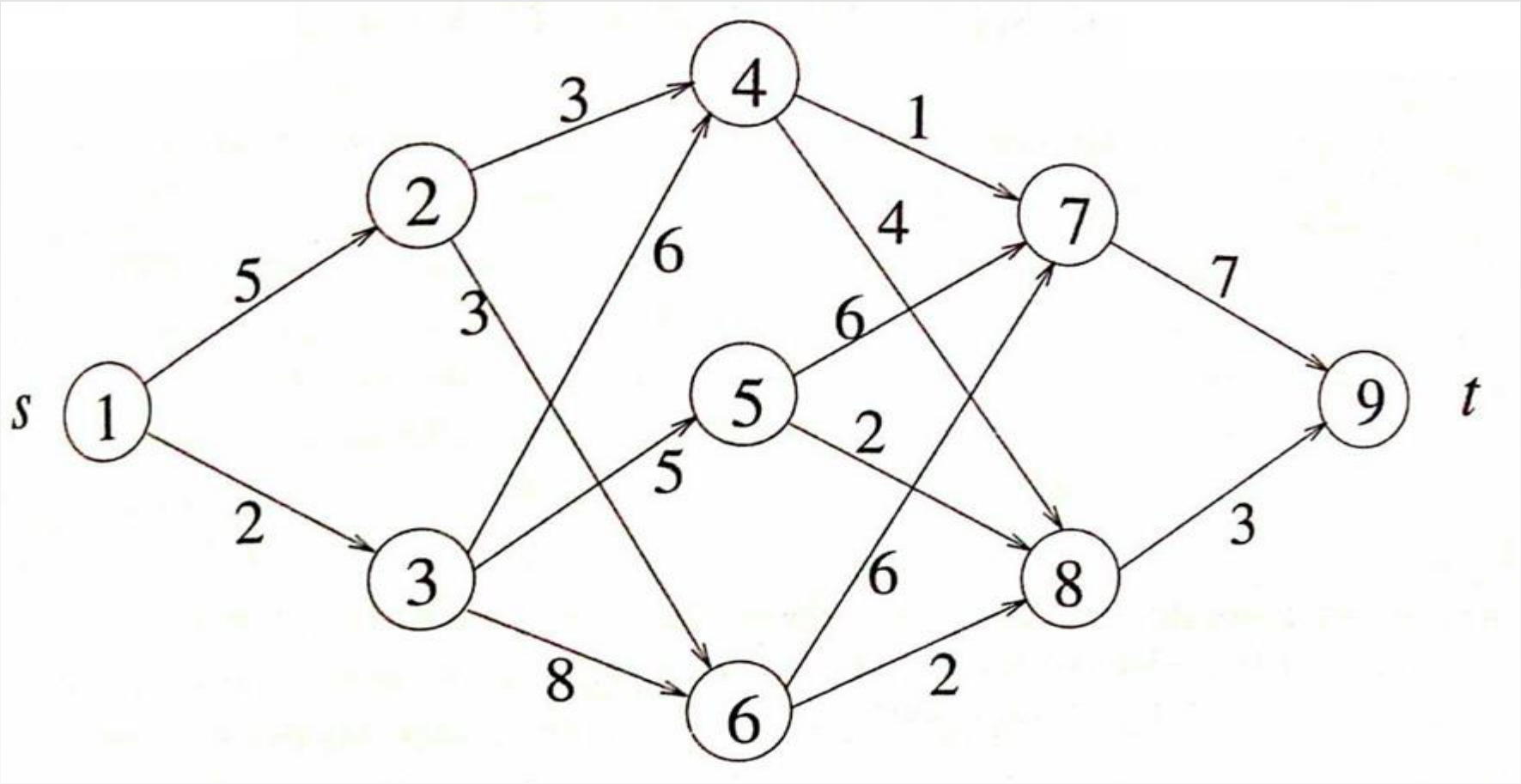
$$cost(3,6) = min\{c(6,9) + cost(4,9), c(6,10) + cost(4,10)\}$$
$$= min\{6 + cost(4,9), 5 + cost(4,10)\}$$
$$= 7$$
$$cost(3,7) = min\{4 + cost(4,9), 3 + cost(4,10)\}$$
$$= 5$$
$$cost(3,8) = 7$$
$$cost(2,2) = min\{4 + cost(3,6), 2 + cost(3,7), 1 + cost(3,8)\}$$
$$= 7$$
$$cost(2,3) = 9$$
$$cost(2,4) = 18$$
$$cost(2,5) = 15$$
$$cost(1,1)$$
$$= min\{9 + cost(2,2), 7 + cost(2,3), 3 + cost(2,4),$$
$$2 + cost(2,5)\}$$
$$= 16$$

```
1.  Algorithm FGraph (G,k,n,p)
2.  //  The Input is a k-stage graph G=(V,E) with 'n' vertices
3.  //  indexed in order of stages.  E is a set of edges and
4.  // c[i,j] is the cost of <i,j>. p[1:k] is a minimum cost path.
5.  {
6.       cost[n]:= 0.0;
7.       for j:= n − 1 to 1 step − 1 do
8.    { // Compute cost[j],
9.     Let 'r' be the vertex such that < j,r > is an edge of
10.    'G' and c[j,r] + cost[r] is minimum.
11.     cost[j] = c[j + r] + cost[r];
12.     d[j] = r;
13.   }
14.   // Find a minimum cost path.
15.       P[1] = 1;       P[k] = n;
16.       for j = 2 to k − 1 do
17.       P[j] = d[p[j − 1]];
18. }
```

Multistage Graphs Exercise

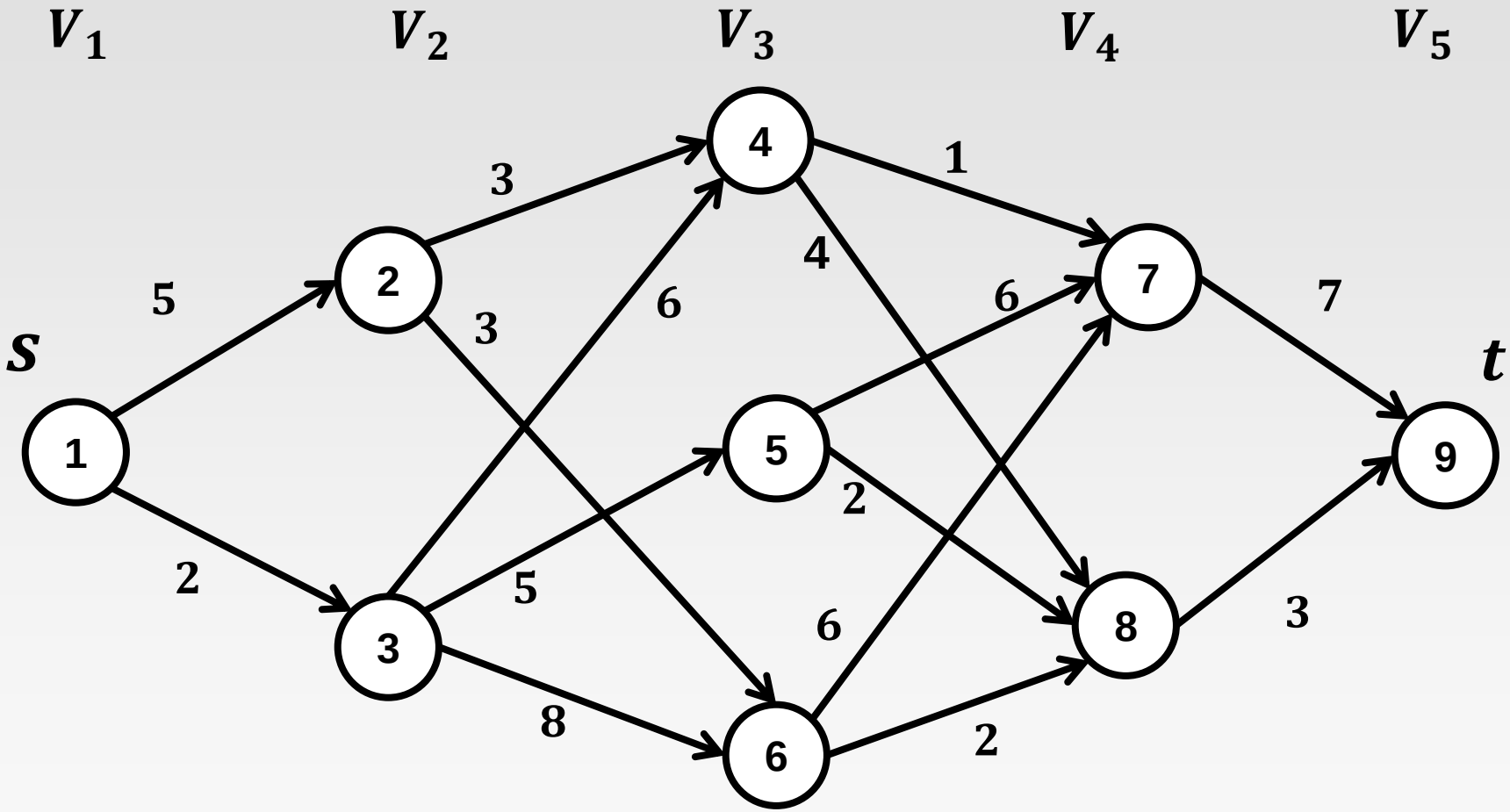
- Find a minimum-cost path from 's' to 't' using Forward approach and then by using Backward Approach.

V_1 V_2 V_3 V_4 V_5



Multistage Graphs Exercise

- Find a minimum-cost path from 's' to 't' using Forward approach and then by using Backward Approach.



0/1 KNAPSACK

- Already we have seen a solution to Knapsack Problem using Greedy Method.
- Here we are going to solve 0/1 Knapsack Problem using Dynamic Programming.
- This problem is similar to ordinary Knapsack Problem but we may not take a fraction of an object.
- We are given 'n' objects with weight w_i and profits p_i where 'i' varies from 1 to n and also a knapsack with capacity 'm'.
- The problem is, we have to fill the bag with the help of 'n' objects and the resulting Profit has to be maximum.
- A solution to the knapsack problem can be obtained by making sequence of decisions on the variables $x_1, x_2 \ldots\ldots x_n$

0/1 KNAPSACK

- A decision on variable x_i involves determining which of the values 0 *or* 1 is to be assigned to it.
- Formally the problem can be stated as

$$\textbf{Maximize} \sum_{1 \le i \le n} p_i x_i$$

$$\textbf{Subject to} \sum_{1 \le i \le n} w_i x_i \le m$$

$$\textbf{and } x_i \in \{0, 1\}, \qquad 1 \le i \le n$$

- Let us assume that decisions on the x_i are made in the order of $x_1, x_2 \ldots\ldots x_n$.
- Following a decision on x_i, we may be in one of the two possible states : the capacity remaining in the knapsack is m and no profit has accrued or the capacity remaining is $m - w_i$ and a profit of p_i has accrued.

0/1 KNAPSACK

- Let $g_0(m)$ be the value of an optimal solution to Knapsack.
- Since the principle of optimality holds, we obtain

$$g_0(m) = \max\{ g_1(m), g_1(m - w_1) + p_1 \}$$

- $g_0(m)$ will be Maximum Profit for 0/1 Knapsack.

1. $g_1(m)$ will be choice one where we are not adding 1st object so no profit earned.
2. $g_1(m - w_1) + p_1$ will be choice two where we are adding 1st object so p_1 profit earned and for adding next objects remaining capacity will be $m - w_1$.

- For arbitrary $g_i(y), i > 0$ the above equation generalizes to

$$g_i(y) = \max\{ g_{i+1}(y), g_{i+1}(y - w_{i+1}) + p_{i+1} \}$$

- Consider the knapsack instance $n = 3, m = 6$

$(w_1, w_2, w_3) = (2, 3, 4)$ and $(p_1, p_2, p_3) = (1, 2, 5)$

➢ We will start with first element using formula

$$g_0(m) = \max\{ g_1(m), g_1(m - w_1) + p_1 \}$$

$$g_0(6) = \max\{ g_1(6), g_1(6 - 2) + 1 \}$$

$$\boxed{g_0(6) = \max\{ g_1(6), g_1(4) + 1 \}} \qquad \text{Eq.1}$$

➢ We will now find value of $g_1(6)$

$$g_1(6) = \max\{ g_2(6), g_2(6 - 3) + 2 \}$$

$$g_1(6) = \max\{ g_2(6), g_2(3) + 2 \}$$

➢ Now find value of $g_2(6)$

$$g_2(6) = \max\{ g_3(6), g_3(6 - 4) + 5 \}$$

$$g_2(6) = \max\{ g_3(6), g_3(2) + 5 \} = max\{ 0, 0 + 5 \} = 5$$

- Now find value of $g_2(3)$

$$g_2(3) = \max\{\, g_3(3),\ g_3(3-4) + 5 \,\}$$

$$g_1(6) = \max\{0, -\infty + 5\} = 0$$

- Therefore, value of $g_1(6)$ from Eq 1.

$$\boxed{g_1(6) = \max\{5, 0 + 2\} = 5} \qquad \text{Eq.2}$$

- We will now find value of $g_1(4)$

$$g_1(4) = \max\{\, g_2(4),\ g_2(4-3) + 2 \,\}$$

$$g_1(4) = \max\{\, g_2(4),\ g_2(1) + 2 \,\}$$

- Now find value of $g_2(4)$

$$g_2(4) = \max\{\, g_3(4),\ g_3(4-4) + 5 \,\}$$

$$g_2(4) = \max\{0, 0 + 5\} = 5$$

- Now find value of $g_2(1)$

$$g_2(1) = \max\{\, g_3(1),\ g_3(1-4) + 5 \,\}$$

$$g_1(1) = \max\{0, -\infty + 5\} = 0$$

- Therefore, value of $g_1(4)$ from Eq 1.

$$\boxed{g_1(4) = \max\{5, 0 + 2\} = 5} \qquad \text{Eq. 3}$$

- We will now find final value of $g_0(6)$ using values of Eq. 2 & Eq. 3 by putting in Eq.1

$$g_0(6) = \max\{\, g_1(6),\ g_1(4) + 1 \,\} \qquad \text{Eq.1}$$

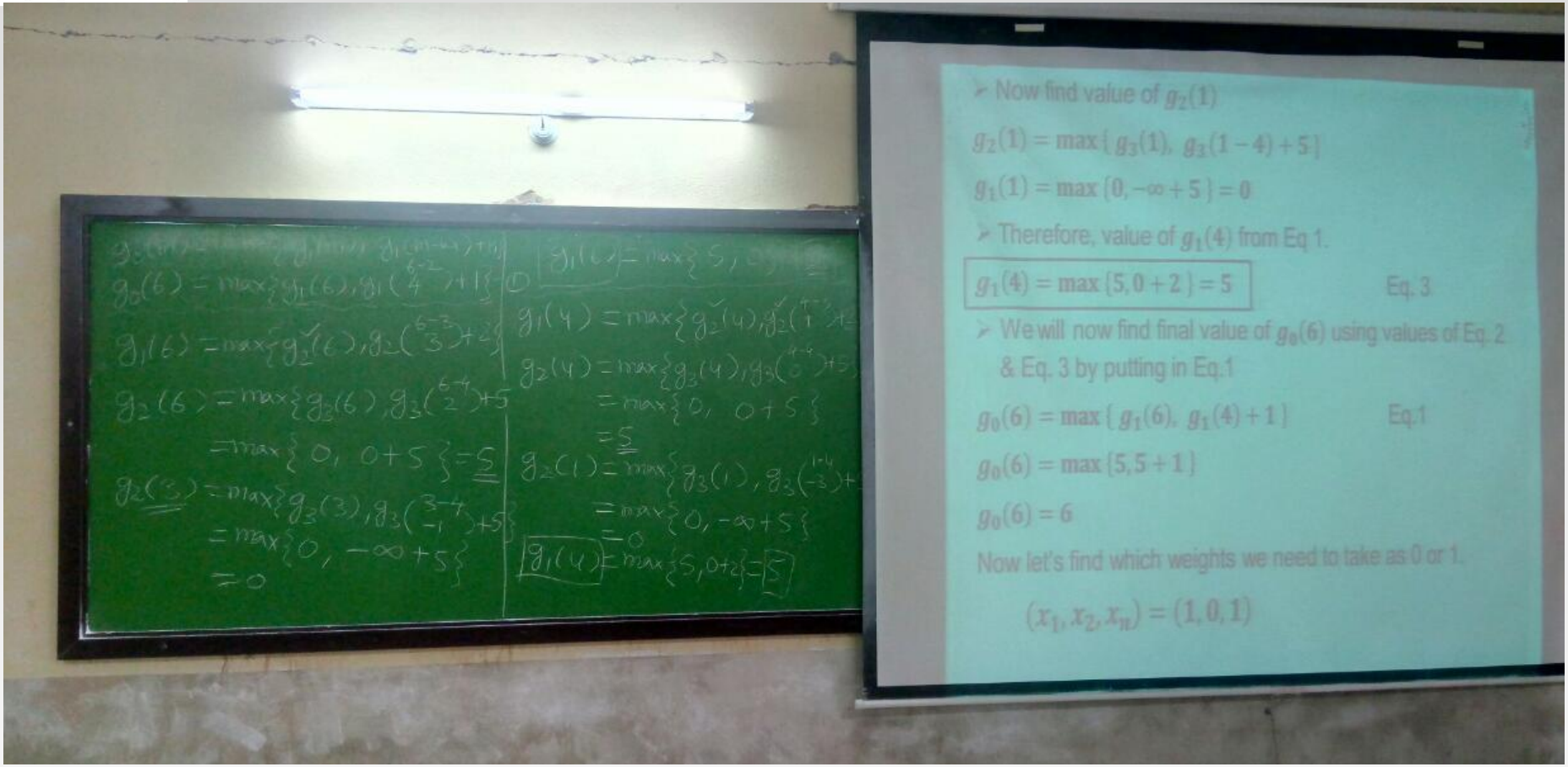
$$g_0(6) = \max\{5, 5 + 1\}$$

$$\boxed{g_0(6) = 6}$$

Now let's find which weights we need to take as 0 or 1.

$$\boxed{(x_1, x_2, x_n) = (1, 0, 1)}$$

0/1 KNAPSACK



0/1 Knapsack Problem – On board Solution done on 12-02-20

0/1 Knapsack Exercise

- Consider the following 0/1 Knapsack Problem instances.
- Find maximum profit earned using Dynamic Programming.
- Clearly mention all Steps.
- $n = 3, m = 8$

  $(w_1, w_2, w_3) = (2, 6, 8)$ and $(p_1, p_2, p_3) = (5, 4, 10)$
- $n = 4, m = 8$

$(w_1, w_2, w_3, w_4) = (2, 3, 4, 5)$ and $(p_1, p_2, p_3, p_4) = (1, 2, 5, 6)$
- $n = 4, m = 8$

$(w_1, w_2, w_3, w_4) = (1, 5, 3, 4)$ and

$(p_1, p_2, p_3, p_4) = (15, 10, 9, 5)$

Travelling Salesperson Problem (TSP)

- We have seen how to find solution to 0/1 Knapsack Problem using Dynamic Programming, which is subset problem.
- Now we will work on a permutation problem.
- Permutation problems are usually harder to solve than subset problems as there are $n!$ different permutations of n objects
- Let $G = (V, E)$ be a directed graph with edge cost c_{ij}.
- The cost c_{ij} is defined such that $c_{ij} > 0$ for all i and j and $c_{ij} = \infty$, if $\langle i, j \rangle \notin$ **E**.
- Let $|V| = n$ (i.e. number of nodes) and assume $n > 1$
- A tour of G is a directed cycle that include every vertex in V.
- The traveling salesperson problem is to find a tour of minimum cost.

Travelling Salesperson Problem (TSP)

- The cost of the tour is the sum of cost of the edges on the tour.
- The tour is the shortest path that starts and ends at the same vertex (i.e.1).
- Suppose we have to route a postal van to pick up mail from the mail boxes located at 'n' different sites.
- An $n + 1$ vertex graph can be used to represent the situation.
- One vertex represent the post office from which the postal van starts and return.
- Edge $\langle i, j \rangle$ is assigned a cost equal to the distance from site 'i' to site 'j'.
- The route taken by the postal van is a tour and we are interested in finding a tour of minimum length.

Travelling Salesperson Problem (TSP)

- Every tour consists of an edge $\langle 1, k \rangle$ for some $k \in V - \{1\}$ and a path from vertex k to vertex 1.
- The path from vertex k to vertex 1 goes through each vertex in $V - \{1, k\}$ exactly once.
- The function which is used to find the path is

$$g(1, V - \{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V - \{1, k\})\}$$

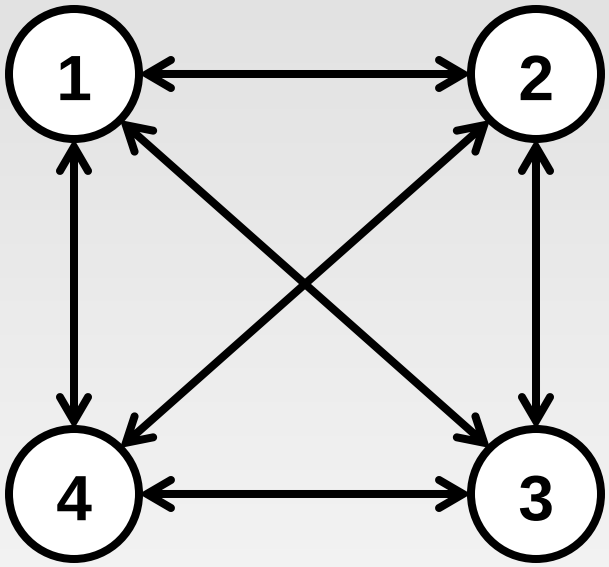
Generalizing this equation we obtain (for $i \notin S$)

$$g(i, S) = \min_{j \in S} \{c_{ij} + g(j, S - \{j\})\}$$

- $g(i, S)$ be the length of a shortest path starting at vertex i, going through all vertices in S, and terminating at vertex 1.
- The function $g(1, V - \{1\})$ is the length of an optimal tour.

Travelling Salesperson Problem (TSP)

- Consider the directed graph. The edge lengths are given by matrix c.



(a)

c	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

(b)

Directed Graph and Edge Length Matrix c

- We will start with finding minimum length starting from first node to all other nodes and back to first node using formula

$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V-\{1,k\})\}$$

$$g(1,\{2,3,4\}) = min \begin{Bmatrix} c_{12} + g(2,\{3,4\}), \\ c_{13} + g(3,\{2,4\}), \\ c_{14} + g(4,\{2,3\}) \end{Bmatrix}$$

- Let's find all values one by one, starting with $g(2,\{3,4\})$

$$g(2,\{3,4\}) = min \begin{Bmatrix} c_{23} + g(3,\{4\}), \\ c_{24} + g(4,\{3\}) \end{Bmatrix}$$

- Now find $g(3,\{4\})$

$$g(3,\{4\}) = min\{c_{34} + g(4,\{\emptyset\})\}$$

Note :- $(4,\{\emptyset\})$ **indicates reaching back to Source Node i.e. to 1.**

$$= c_{34} + c_{41} = 12 + 8 = 20$$

- Now find $g(4,\{3\})$

$$g(4,\{3\}) = min\{c_{43} + g(3,\{\emptyset\})\}$$
$$= c_{43} + c_{31} = 9 + 6 = 15$$

Therefore

$$g(2,\{3,4\}) = min\begin{Bmatrix} c_{23} + g(3,\{4\}), \\ c_{24} + g(4,\{3\}) \end{Bmatrix} = min\begin{Bmatrix} 9 + 20, \\ 10 + 15 \end{Bmatrix} = 25$$

- Now let's find value of $g(3,\{2,4\})$

$$g(3,\{2,4\}) = min\begin{Bmatrix} c_{32} + g(2,\{4\}), \\ c_{34} + g(4,\{2\}) \end{Bmatrix}$$

- Now find $g(2,\{4\})$

$$g(2,\{4\}) = min\{c_{24} + g(4,\{\emptyset\})\}$$
$$= c_{24} + c_{41} = 10 + 8 = 18$$

- & find $g(4,\{2\})$

$$g(4,\{2\}) = min\{c_{42} + g(2,\{\emptyset\})\}$$
$$= c_{42} + c_{21} = 8 + 5 = 13$$

Therefore

$$g(3,\{2,4\}) = min\begin{Bmatrix} c_{32} + g(2,\{4\}), \\ c_{34} + g(4,\{2\}) \end{Bmatrix} = min\begin{Bmatrix} 13 + 18, \\ 12 + 13 \end{Bmatrix} = 25$$

- Now let's find value of $g(4,\{2,3\})$

$$g(4,\{2,3\}) = min\begin{Bmatrix} c_{42} + g(2,\{3\}), \\ c_{43} + g(3,\{2\}) \end{Bmatrix}$$

- Now find $g(2,\{3\})$

$$g(2,\{3\}) = min\{c_{23} + g(3,\{\emptyset\})\}$$

$$= c_{23} + c_{31} = 9 + 6 = 15$$

- & find $g(3,\{2\})$

$$g(3,\{2\}) = min\{c_{32} + g(2,\{\emptyset\})\}$$

$$= c_{32} + c_{21} = 13 + 5 = 18$$

Therefore

$$g(4,\{2,3\}) = min\begin{Bmatrix} c_{42} + g(2,\{3\}), \\ c_{43} + g(3,\{2\}) \end{Bmatrix} = min\begin{Bmatrix} 8 + 15 \\ 9 + 18 \end{Bmatrix} = 23$$

- Finally we will put all calculated values in original Formula to find value of $g(1,\{2,3,4\})$

$$g(1,\{2,3,4\}) = min \begin{Bmatrix} c_{12} + g(2,\{3,4\}), \\ c_{13} + g(3,\{2,4\}), \\ c_{14} + g(4,\{2,3\}) \end{Bmatrix}$$

$$= min \begin{Bmatrix} 10 + 25, \\ 15 + 25, \\ 20 + 23 \end{Bmatrix}$$

$$= min \begin{Bmatrix} 35, \\ 40, \\ 43 \end{Bmatrix} = 35$$

- **So the minimum length i.e. optimal tour of Graph from Node 1 is 35.**
- Now let's find the optimal tour path from 1.

$$1 - 2 - 4 - 3 - 1$$

All-Pairs Shortest Paths

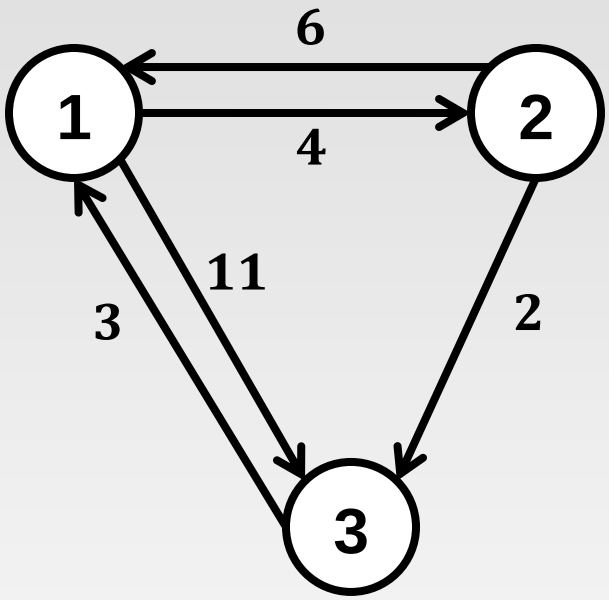
- Let $G = (V, E)$ be a directed graph n vertices.
- Let cost be a $cost$ adjacency matrix for G such that
  - $cost(i, i) = 0, 1 \leq i \leq n$ i.e. distance to same node is 0.
  - $cost(i, j)$ is the length or cost of edge $\langle i, j \rangle$, if $\langle i, j \rangle \epsilon E(G)$
  - and $cost(i, j) = \infty$ if $i \neq j$ and $\langle i, j \rangle \notin E(G)$.
- The **All-Pairs Shortest Paths** problem is to determine a matrix A such that $A(i, j)$ is length of shortest path from i to j.
- The matrix A can be obtained by solving n **Single Source Shortest Path** Problem using **ShorestPath** algorithm (Unit III) to all the nodes, with time complexity $O(n^3)$.

➢ Using Dynamic Programming, we can obtain alternate solution using formula,

$$A^k(i, j) = \min_{1 \leq k \leq n} \{A^{k-1}(i, j), A^{k-1}(i, k) + A^{k-1}(k, j)\}$$

All-Pairs Shortest Paths

- Consider the directed graph. The edge lengths are given by matrix A^0.



$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array}$$

(a) **(b)**

Directed Graph and Edge Length Matrix A^0

All-Pairs Shortest Paths

- We will start with finding minimum length from each node to all other nodes using formula

$$A^k(i,j) = \min_{1 \le k \le n} \left\{ A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j) \right\}$$

- We will start with finding minimum length from all nodes at Level-I

$$A^1(1,2) = min\left\{A^0(1,2), A^0(1,1) + A^0(1,2)\right\}$$
$$= min\{4, 0+4\} = 4$$

$$A^1(1,3) = min\left\{A^0(1,3), A^0(1,1) + A^0(1,3)\right\}$$
$$= min\{11, 0+11\} = 11$$

$$A^1(2,1) = min\left\{A^0(2,1), A^0(2,1) + A^0(1,1)\right\}$$
$$= min\{6, 6+0\} = 6$$

All-Pairs Shortest Paths

$$A^1(2,3) = min\{A^0(2,3), A^0(2,1) + A^0(1,3)\}$$
$$= min\{2, 6 + 11\} = 2$$
$$A^1(3,1) = min\{A^0(3,1), A^0(3,1) + A^0(1,1)\}$$
$$= min\{3, 3 + 0\} = 3$$
$$A^1(3,2) = min\{A^0(3,2), A^0(3,1) + A^0(1,2)\}$$
$$= min\{\infty, 3 + 4\} = 7$$

- Therefore the cost matrix at Level-I is

$$A^1 \quad \begin{matrix} & 1 & 2 & 3 \\ 1 & 0 & 4 & 11 \\ 2 & 6 & 0 & 2 \\ 3 & 3 & 7 & 0 \end{matrix}$$

All-Pairs Shortest Paths

- Now we will find minimum length from all nodes at Level-II

$$A^2(1,2) = min\{A^1(1,2), A^1(1,2) + A^1(2,2)\}$$

$$= min\{4, 4+0\} = 4$$

$$A^2(1,3) = min\{A^1(1,3), A^1(1,2) + A^1(2,3)\}$$

$$= min\{11, 4+2\} = 6$$

$$A^2(2,1) = min\{A^1(2,1), A^1(2,2) + A^1(2,1)\}$$

$$= min\{6, 0+6\} = 6$$

$$A^2(2,3) = min\{A^1(2,3), A^1(2,2) + A^1(2,3)\}$$

$$= min\{2, 0+2\} = 2$$

All-Pairs Shortest Paths

$$A^2(3,1) = min\{A^1(3,1), A^1(3,2) + A^1(2,1)\}$$
$$= min\{3, 7+6\} = 3$$
$$A^2(3,2) = min\{A^1(3,2), A^1(3,2) + A^1(2,2)\}$$
$$= min\{7, 7+0\} = 7$$

- Therefore the cost matrix at Level-II is

$$A^2 \quad \begin{matrix} & 1 & 2 & 3 \\ 1 & 0 & 4 & 6 \\ 2 & 6 & 0 & 2 \\ 3 & 3 & 7 & 0 \end{matrix}$$

All-Pairs Shortest Paths

- Now we will find minimum length from all nodes at Level-III

$$A^3(1,2) = min\{A^2(1,2), A^2(1,3) + A^2(3,2)\}$$
$$= min\{4, 6+7\} = 4$$

$$A^3(1,3) = min\{A^2(1,3), A^2(1,3) + A^2(3,3)\}$$
$$= min\{6, 6+0\} = 6$$

$$A^3(2,1) = min\{A^2(2,1), A^2(2,3) + A^2(3,1)\}$$
$$= min\{6, 2+3\} = 5$$

$$A^3(2,3) = min\{A^2(2,3), A^2(2,3) + A^2(3,3)\}$$
$$= min\{2, 2+0\} = 2$$

All-Pairs Shortest Paths

$$A^3(3,1) = min\{A^2(3,1), A^2(3,3) + A^1(3,1)\}$$
$$= min\{3, 0+3\} = 3$$
$$A^3(3,2) = min\{A^2(3,2), A^2(3,3) + A^2(3,2)\}$$
$$= min\{7, 0+7\} = 7$$

- Therefore the cost matrix at Level-II is

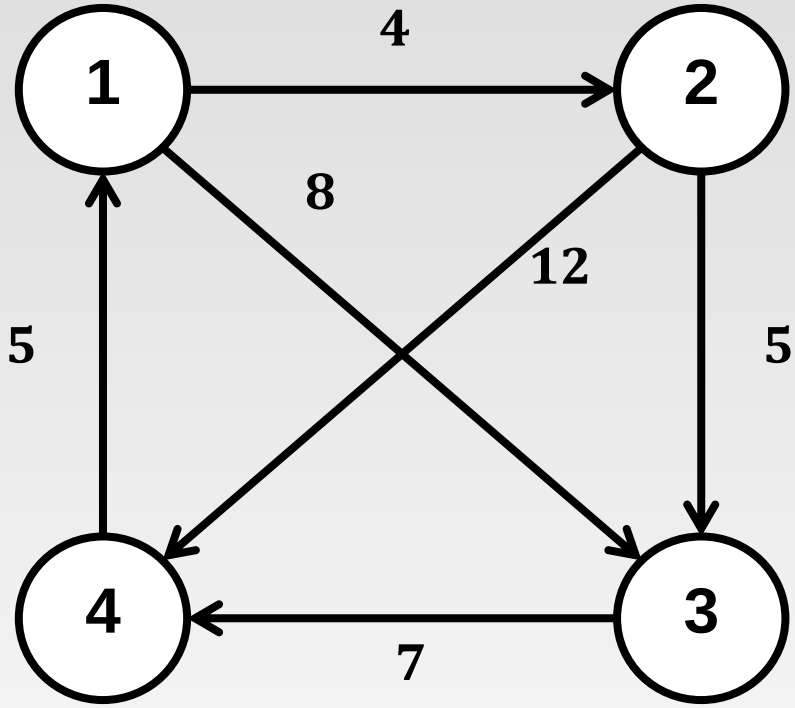
$$A^3 \quad \begin{matrix} 1 & 2 & 3 \end{matrix}$$
$$\begin{matrix} 1 \\ 2 \\ 3 \end{matrix} \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Algorithm to compute lengths of Shortest Paths

```
1.  Algorithm AllPaths (cost, A, n)
2.  // cost[ 1:n, 1:n ] is the cost adjacency matrix of a graph
3.  // with n vertices; A[i,j]is the cost of a shortest path from
4.  // vertex i to vertex j. cost[i,i]=0.0, for 1≤i≤n.
5.  {
6.     for i := 1 to n do
7.           for j := 1 to n do
8.              A[i, j] := cost[i, j];
9.     for k := 1 to n do
10.       for i := 1 to n do
11.          for j := 1 to n do
12.             A[i, j] := min(A[i, j], A[i, k] + A[k, j]);
13. }
```

All-Pairs Shortest Paths Exercise

- Compute all pair shortest path for following.



$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 4 & 8 & \infty \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

Hint/Observation :

- **If $i = k$ or $j = k$ then $A^k(i, j)$ remains same, no need to re-calculate.**
- **There will not be any change of Value.**

All-Pairs Shortest Paths Exercise

A^1

	1	2	3	4
1	0	4	8	∞
2	∞	0	5	12
3	∞	∞	0	7
4	5	9	13	0

A^2

	1	2	3	4
	0	4	8	16
	∞	0	5	12
	∞	∞	0	7
	5	9	13	0

A^3

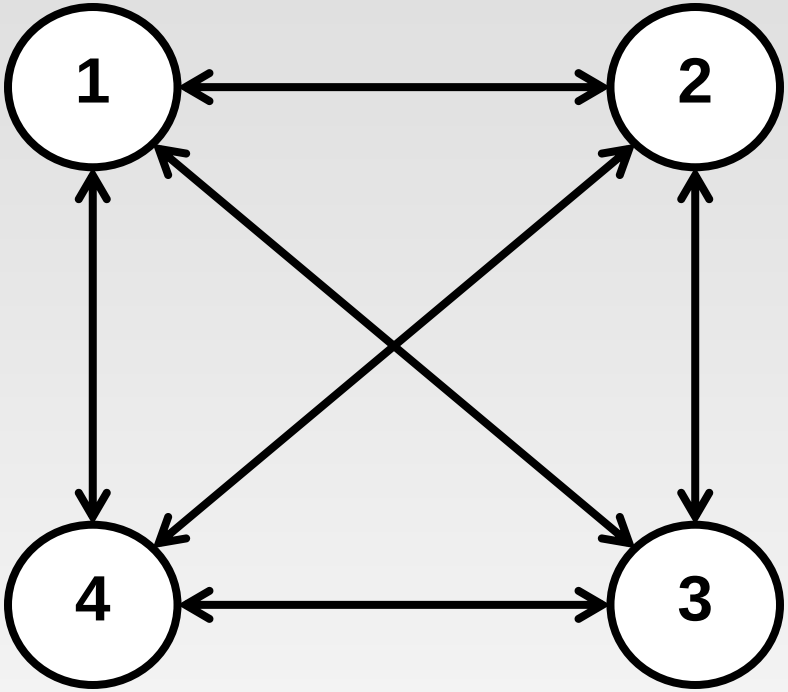
	1	2	3	4
1	0	4	8	15
2	∞	0	5	12
3	∞	∞	0	7
4	5	9	13	0

A^4

	1	2	3	4
	0	4	8	15
	17	0	5	12
	12	16	0	7
	5	9	13	0

All-Pairs Shortest Paths Exercise

- Compute all pair shortest path for following.



$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix} \end{array}$$

Hint/Observation :

- **If $i = k$ or $j = k$ then $A^k(i, j)$ remains same, no need to re-calculate.**
- **There will not be any change of Value.**

Matrix Chain Multiplication

- Matrix multiplication : Let $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ are three matrices such that $\boldsymbol{C} = \boldsymbol{A} \times \boldsymbol{B}$.
- Let dimensions of $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{C}$ are $\boldsymbol{m} \times \boldsymbol{n},\ \boldsymbol{n} \times \boldsymbol{p}\ \boldsymbol{and}\ \boldsymbol{m} \times \boldsymbol{p}$
- From the definition of matrix multiplication,

$$\boldsymbol{C(i,j)} = \sum_{\boldsymbol{1}}^{\boldsymbol{n}} \boldsymbol{A(i,k)\ B\ (k,j)}$$

- The number of element multiplications required for this algorithm is $\boldsymbol{m} \times\ \boldsymbol{n}\ \times \boldsymbol{p}.$
- Let $\boldsymbol{M_1} \times \boldsymbol{M_2} \times \boldsymbol{M_3} \times \ldots\ldots\ \boldsymbol{M_r}$ be a chain of matrix products.
- This chain can be evaluated in several different ways.
  - Ex : $\left(\ldots\left(\left((\boldsymbol{M_1} \times\ \boldsymbol{M_2}) \times \boldsymbol{M_3}\right) \times \boldsymbol{M_4}\right) \times \cdots . .\right) \times \boldsymbol{M_r}$ or
  - $\left(\boldsymbol{M_1} \times \left(\boldsymbol{M_2} \times \left(\boldsymbol{M_3} \times \left(\ldots . . \times \left(\boldsymbol{M_{r-1}} \times \boldsymbol{M_r}\right)\right)\right)\right) \ldots .\right)$
  - etc.

Matrix Chain Multiplication

- The cost of any computation of $\boldsymbol{M_1 \times M_2 \times M_3 \times \ldots\ldots M_r}$ is the number of multiplications used.
- The Matrix Chain Multiplication Problem is to optimal cost i.e. minimum number of multiplications required to compute a chain of matrix products i.e. $\boldsymbol{M_1 \times M_2 \times M_3 \times \ldots\ldots M_r}$.

- Consider $\boldsymbol{r = 4}$ i.e. chained multiplication of 4 matrices $\boldsymbol{M_1, M_2, M_3, and\ M_4}$ with dimensions $\boldsymbol{100 \times 1,\ 1 \times 100, 100 \times 1}$ **and** $\boldsymbol{1 \times 100}$ respectively.
- This chained multiplication of these 4 matrices can be computed using five ways.

Matrix Chain Multiplication

- Here $r = 4$ and matrix dimensions are : $M_1(100 \times 1)$, $M_2(1 \times 100)$, $M_3(100 \times 1)$, $and\ M_4(1 \times 100)$. We need to find Optimal Cost of Multiplication.

- $I\ Way = \left(((M_1 \times M_2) \times M_3) \times M_4\right)$

$(M_1 \times M_2) = ((100 \times 1 \times 100)) = 10,000$ Multiplications & size of resultant matrix is 100×100

$((M_1 \times M_2) \times M_3) = (100 \times 100 \times 1) = 10,000$ Multiplications & size of resultant matrix is 100×1

$\left(((M_1 \times M_2) \times M_3) \times M_4\right) = (100 \times 1 \times 100) = 10,000$ Multiplications & size of resultant matrix is 100×100.

- So total multiplications required are

$$\text{No of Multiplication} = 10,000 + 10,000 + 10,000 = 30,000$$

Matrix Chain Multiplication

- $\boldsymbol{II\ Way = \left((M_1 \times\ M_2) \times (M_3 \times\ M_4)\right)}$

$\boldsymbol{(M_1 \times\ M_2)\ = \left((100 \times\ 1\ \times 100)\right) = 10,000}$ Multiplications & size of resultant matrix is $\boldsymbol{100\ \times 100}$

$\boldsymbol{(M_3 \times\ M_4)\ = \left((100 \times\ 1\ \times 100)\right) = 10,000}$ Multiplications & size of resultant matrix is $\boldsymbol{100\ \times 100}$

$\boldsymbol{\left((M_1 \times\ M_2) \times (M_3 \times\ M_4)\right)\ = (100 \times\ 100 \times 100) = 10,00,000}$ Multiplications & size of resultant matrix is $\boldsymbol{100\ \times 100}$

- So total multiplications required are

$$\textbf{No of Multiplication} = 10,000 + 10,000 + 10,00,000$$
$$= 10,20,000$$

Matrix Chain Multiplication

- $\boldsymbol{III\ Way = \left(M_1 \times \left((M_2 \times M_3) \times M_4\right)\right)}$

$\boldsymbol{(M_2 \times M_3) = \left((1 \times 100 \times 1)\right) = 100}$ Multiplications & size of resultant matrix is $\boldsymbol{1 \times 1}$

$\boldsymbol{\left((M_2 \times M_3) \times M_4\right) = \left((1 \times 1 \times 100)\right) = 100}$ Multiplications & size of resultant matrix is $\boldsymbol{1 \times 100}$

$\boldsymbol{\left(M_1 \times \left((M_2 \times M_3) \times M_4\right)\right) = (100 \times 1 \times 100) = 10,000}$ Multiplications & size of resultant matrix is $\boldsymbol{100 \times 100}$

- So total multiplications required are

$\boldsymbol{\text{No of Multiplication} = 100 + 100 + 10,000 = 10,200}$

Matrix Chain Multiplication

- $\boldsymbol{IV\ Way = ((M_1 \times (M_2 \times M_3)) \times M_4)}$

$\boldsymbol{(M_2 \times M_3) = ((1 \times 100 \times 1)) = 100}$ Multiplications & size of resultant matrix is $\boldsymbol{1 \times 1}$

$\boldsymbol{(M_1 \times (M_2 \times M_3)) = ((100 \times 1 \times 1)) = 100}$ Multiplications & size of resultant matrix is $\boldsymbol{100 \times 1}$

$\boldsymbol{((M_1 \times (M_2 \times M_3)) \times M_4) = (100 \times 1 \times 100) = 10,000}$ Multiplications & size of resultant matrix is $\boldsymbol{100 \times 100}$

- So total multiplications required are

$\boldsymbol{\text{No of Multiplication} = 100 + 100 + 10,000 = 10,200}$

Matrix Chain Multiplication

- $V\ Way = (M_1 \times (M_2 \times (M_3 \times M_4)))$

$(M_3 \times M_4) = ((100 \times 1 \times 100)) = 10,000$ Multiplications & size of resultant matrix is 100×100

$(M_2 \times (M_3 \times M_4)) = (1 \times 100 \times 100) = 10,000$
Multiplications & size of resultant matrix is 1×100

$(M_1 \times (M_2 \times (M_3 \times M_4))) = (100 \times 1 \times 100) = 10,000$
Multiplications & size of resultant matrix is 100×100.

- So total multiplications required are

$$\textbf{No of Multiplication} = 10,000 + 10,000 + 10,000 = 30,000$$

Matrix Chain Multiplication

- Here is the summary of all 5 Ways of Chain of Matrix Multiplication
- No of Element Multiplications required are…..
  - $\mathbf{I\ Way\ = 30,000}$
  - $\mathbf{II\ Way\ = 10,20,000}$
  - $\mathbf{III\ Way\ = 10,200}$
  - $\mathbf{IV\ Way\ = 10,200}$
  - $\boldsymbol{V\ Way = \ 30,000}$
- From this we can conclude that if we use **3rd or 4th way** then we will get Optimal Number of Multiplications i.e. $\mathbf{10,200}$.
- Whereas the **II way** gives Worst Case i.e. highest number of Multiplications = $\mathbf{10,20,000}$.

Matrix Chain Multiplication Exercise

- Consider $\mathbf{4}$ matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \& \boldsymbol{D}$ with following dimensions

$$\boldsymbol{A} = \mathbf{13} \times \mathbf{5}$$

$$\boldsymbol{B} = \mathbf{5} \times \mathbf{89}$$

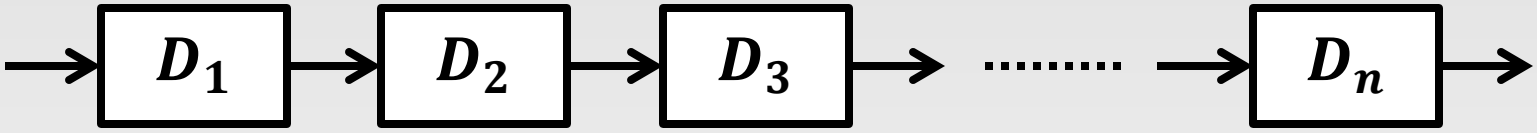
$$\boldsymbol{C} = \mathbf{89} \times \mathbf{3}$$

$$\boldsymbol{D} = \mathbf{3} \times \mathbf{34}$$

- Find the Cost of Multiplication using few Possible ways and also find Optimal Cost of Multiplication.
- **Hint : Optimal Solution is 2,856.**

Reliability Design

- Consider example of how to use Dynamic Programming to solve a problem with Multiplicative Optimization Function.
- The problem is to design a system that is composed of several devices connected in series.



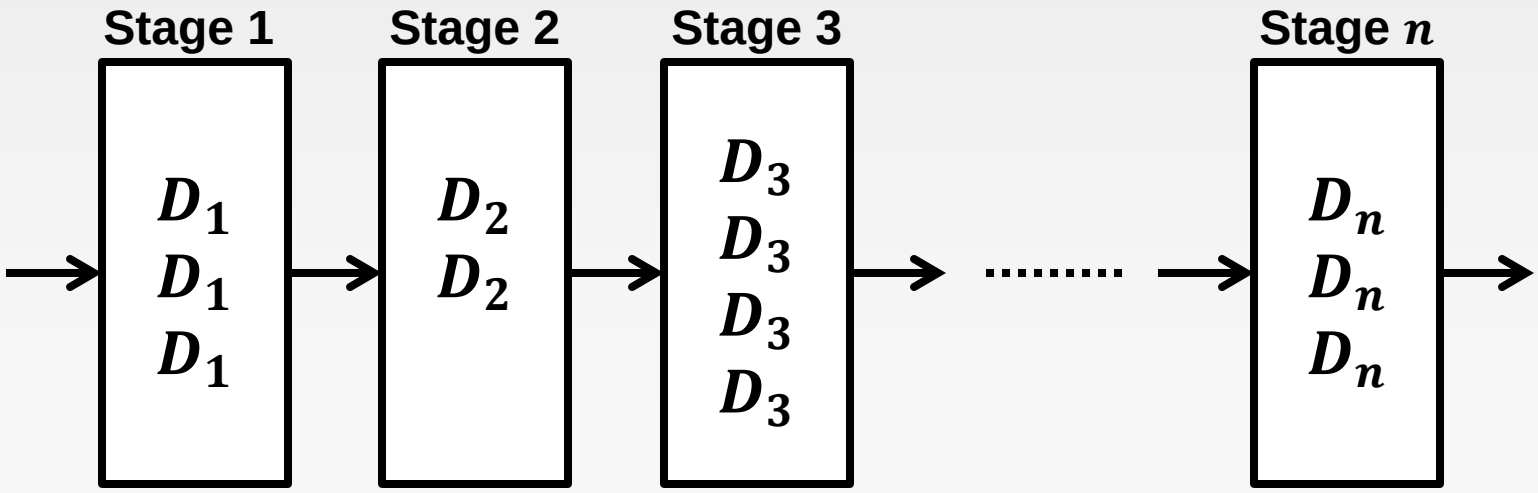
- Let r_i be the reliability of device D (that is, r_i is the probability that device i will function properly).
- As there are n devices connected in series, the reliability of the entire system is $\prod r_i$
- This means even if individual devices are reliable (r_i's are very close to 1), the reliability of the entire system may not be very good.

Reliability Design

- For example, if $n = 10$ and $r_i = 0.99, 1 \le i \le 10$, then

  $\prod r_i = 0.904.$ $(0.99 \times 0.99 \times \cdots \times 0.99)$ 10 times.

- If $n = 3$ and $r_i = 0.9, 1 \le i \le 3$, then $\prod r_i = 0.729.$
- If $n = 4$ and $r_i = 0.9, 1 \le i \le 4$, then $\prod r_i = 0.6591.$
- Hence, to improve the reliability of system, it is desirable to duplicate devices i.e. multiple devices (copies) of the same device type are connected in parallel using switching circuits.



- The switching circuits determine which devices in any given group are functioning properly, use one of working such device at each stage.
- If stage $\boldsymbol{i}$ contains $\boldsymbol{m_i}$ copies of device $\boldsymbol{D_i}$, then probability that all $\boldsymbol{m_i}$ have a malfunction (not working) is $\boldsymbol{(1 - r_i)^{m_i}}$.
- Hence the reliability of stage $\boldsymbol{i}$ becomes $\boldsymbol{1 - (1 - r_i)^{m_i}}$.
- Thus if $\boldsymbol{r_i = 0.99}$ and $\boldsymbol{m_i = 2}$, the stage reliability $= \boldsymbol{0.9999}$.
- In any practical systems, the stage reliability is a little less than $\boldsymbol{1 - (1 - r_i)^{m_i}}$ as switching circuits also not fully reliable.
- Let us assume that the reliability of stage $\boldsymbol{i}$ is given by function $\boldsymbol{\emptyset_i(m_i)}$.
- Then the the reliability of system with $\boldsymbol{n}$ stages is

$$\boldsymbol{\prod_{1 \le i \le n} \emptyset_i(m_i)}$$

- Our problem is to use device duplication to maximize reliability, under the Cost constraint.
- Let c be the maximum cost of entire system and let c_i be the cost of each unit of device i.
- We wish to solve the following maximization problem :

$$\textbf{Maximize } \prod_{1 \le i \le n} \emptyset_i(m_i)$$

$$\textbf{Subject to } \sum_{1 \le i \le n} c_i m_i \le c$$

$$m_i \ge 1 \; and \; 1 \le i \le n$$

- The system must consist of at least (min) one unit of each device type, the upper bound (max) of number of devices at stage i can be found using

$$u_i = \left\lfloor \left(c - \left(\left(\sum_{j=1}^{n} c_j \right) - c_i \right) \right) / c_i \right\rfloor$$

Here $c = Total\ Cost.$ $\quad c_i = Cost\ of\ device\ D_i\ (at\ i^{th} stage)$

$\sum_{j=1}^{n} c_j =$ **Sum of** $Cost\ of\ all\ devices\ -one\ unit\ each.$

$\lfloor \quad \rfloor\ is\ floor\ function\ -gives\ integer\ part\ of\ result$

- The optimal solution to problem is $m_1, m_2, m_3 \ldots\ldots m_n$ - result of sequence of decision, one decision for each m_i.
- Let $f_i(x)$ represent the maximum value of $\prod_{1 \le j \le i} \emptyset_j(m_j)$ subject to constraint $\sum_{1 \le j \le i} c_j m_j \le x$.
- Using principle of optimality, the optimal solution is $f_n(c)$

$$f_n(c) = \max_{i \le m_n \le u_n} \{\emptyset_n(m_n)\, f_{n-1}(c - c_n m_n)\}$$

Here $\emptyset_n(m_n) = \textbf{Reliability of stage } n$ $\quad$ $c = \textbf{Total Cost}$.

$c_n m_n = \textbf{Cost of } m \textbf{ number of device } D_n \ (\textbf{at } n^{th} \textbf{stage})$

- For any $f_i(x), i \ge 1$, this equation generalizes to

$$f_i(x) = \max_{i \le m_i \le u_i} \{\emptyset_i(m_i)\, f_{i-1}(x - c_i m_i)\}$$

- Clearly, $f_0(x) = 1 \ \ for\ all\ x, 0 \le x \le c$ i.e. reliability of system at the beginning is 1 – No devices selected.

Reliability Design

- Let S^i (Set of all possible combinations of device at particular stage with cost) consist of tuples of the form $(f,\ x), where\ f = f_i(x)$. Here f is reliability at stage i & x is cost of devices.

- **Example :** Design a 3 Stage system with devices $D_1, D_2, and\ D_3$ with Costs $\$30, \$15, and\ \$20$ respectively. The cost of entire system is to be no more than $\$105$. The reliability of each device type is $0.9, 0.8, and\ 0.5$ respectively.

By assuming stage i has m_i devices of D_i type in parallel reliability of i^{th} stage is

$$\emptyset_i(m_i) = 1 - (1 - r_i)^{m_i}$$

- Given $c = 105$ $c_1 = 30$ $c_2 = 15$ $c_3 = 20$

  $r_1 = 0.9$ $r_2 = 0.8$ $r_3 = 0.5$

First of all we need to find upper bound for each type of device using formula. $u_i = \left\lfloor \left(c - \left(\left(\sum_{j=1}^{n} c_j \right) - c_i \right) \right) / c_i \right\rfloor$

$$u_1 = \left\lfloor \left(105 - \left((30 + 15 + 20) - 30 \right) \right) / 30 \right\rfloor = \lfloor 70/30 \rfloor = 2$$

$$u_2 = \left\lfloor \left(105 - \left((30 + 15 + 20) - 15 \right) \right) / 15 \right\rfloor = \lfloor 55/15 \rfloor = 3$$

$$u_3 = \left\lfloor \left(105 - \left((30 + 15 + 20) - 20 \right) \right) / 20 \right\rfloor = \lfloor 60/20 \rfloor = 3$$

Beginning with Zero Stage $S^0 = \{(1, 0)\}$ $(Reliability, Cost)$

For stage 1 – If we use one unit of device D_1 ${S_1}^1 = \{(0.9, 30)\}$

If we use two unit of device D_1 ${S_2}^1 = \{(0.99, 60)\}$

Therefore $S^1 = \{(0.9, 30), (0.99, 60)\}$

For stage 2 – If we use one & two units of device $\boldsymbol{D_1}$ and

If we use one unit of device $\boldsymbol{D_2}$ along with $\boldsymbol{D_1}$

$$\boldsymbol{S_1{}^2} = \{ (\mathbf{0.9 \times 0.8, 30 + 15}), (\mathbf{0.99 \times 0.8, 75}) \}$$

$$= \{ (\mathbf{0.72, 45}), (\mathbf{0.792, 75}) \}$$

If we use two units of device $\boldsymbol{D_2}$ along with $\boldsymbol{D_1}$

$$\boldsymbol{S_2{}^2} = \{ (\mathbf{0.9 \times 0.96, 60}), (\mathbf{0.99 \times 0.96, 90}) \}$$

$$= \{ (\mathbf{0.864, 60}), (\mathbf{0.9504, 90}) \}$$

$(\mathbf{0.9504, 90})$ not feasible as $\boldsymbol{D_3}$ can not be included

If we use three units of device $\boldsymbol{D_2}$ along with $\boldsymbol{D_1}$

$$\boldsymbol{S_3{}^2} = \{ (\mathbf{0.9 \times 0.992, 75}) \} = \{ (\mathbf{0.8928, 75}) \}$$

Therefore, combining these we get

$$\boldsymbol{S^2} = \{(\mathbf{0.72, 45}), (\mathbf{0.792, 75}), (\mathbf{0.864, 60}), (\mathbf{0.8928, 75}) \}$$

The tuple $(\mathbf{0.792, 75})$ is dominated by $(\mathbf{0.8928, 75})$, therefore,

$$\boldsymbol{S^2} = \{(\mathbf{0.72, 45}), (\mathbf{0.864, 60}), (\mathbf{0.8928, 75}) \}$$

For stage 3 – If we use one unit of device D_3 along with D_1, D_2

$S_1^3 = \{(0.72 \times 0.5, 45 + 20), (0.864 \times 0.5, 80), (0.8928 \times 0.5, 95)\}$

$= \{ (0.36, 65), (0.432, 80), (0.4464, 95) \}$

If we use two units of device D_3 along with D_1, D_2

$S_2^3 = \{(0.72 \times 0.75, 85), (0.864 \times 0.75, 100), (0.8928 \times 0.75, 115)\}$

$(0.8928 \times 0.75, 115)$ not feasible as Cost is more than 105.

$S_2^3 = \{ (0.54, 85), (0.648, 100) \}$

If we use three units of device D_3 along with D_1, D_2

$S_3^3 = \{(0.72 \times 0.875, 105), (0.864 \times 0.875, 120), (0.8928 \times 0.875, 115)\}$

$(0.864 \times 0.875, 120), (0.8928 \times 0.875, 115)$ are not feasible as Cost is more than 105.

$S_3^3 = \{(0.72 \times 0.875, 105)\} = \{(0.63, 105)\}$

Therefore, combining these we get

$$S^3 = \begin{Bmatrix} (0.36, 65), (0.432, 80), (0.4464, 95), \\ (0.54, 85), (0.648, 100), (0.63, 105) \end{Bmatrix}$$

- We observe that with cost 85 we are getting 0.54 reliability is better than compared to 95 cost with 0.4464 reliability &
- With cost 100 we are getting 0.648 reliability is better than compared to 105 cost with 0.63 reliability.

Therefore, we will get final $S^3 = \begin{Bmatrix} (0.36, 65), (0.432, 80), \\ (0.54, 85), (0.648, 100) \end{Bmatrix}$

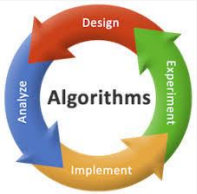
- The best design has a reliability of 0.648 and a cost of 100.
- Tracing back through the S^i, we determine that

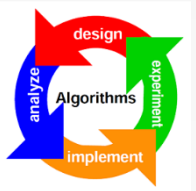
$$m_1 = 1, \qquad m_2 = 2, \qquad m_3 = 2$$

Dynamic Programming - Summary

Using Dynamic Programming, we have seen solution to following problems :

1. Multistage Graphs
2. 0/1 Knapsack Problem
3. Travelling Sales Person Problem
4. All Pairs Shortest Path Problem
5. Matrix Chain Multiplication
6. Reliability Design





VISHNU
UNIVERSAL LEARNING

Department of Information Technology
IT

Next - Unit V

Backtracking

Branch and Bound