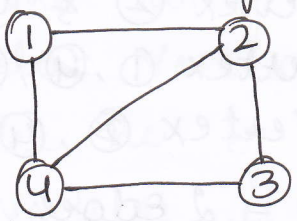
Chapter-4 Dynamic Programming.

Graph is a finite set of vertices (or) nodes (or) points.

Types of Graphs

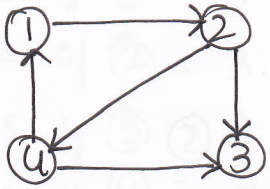
1. Un-Directed Graph.

The vertices which are connected by lines which doesnot have the direction is known as undirected graph. (2-way graph).



2. Directed Graph.

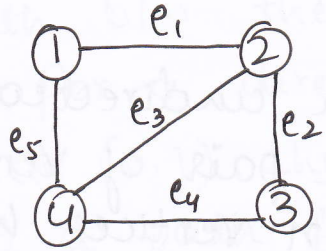
2m The vertices which are connected by arrows and it is directed from one vertex to another vertex is known as directed graph (or) One way graph (Di-graph).



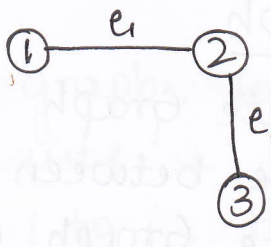
3. Sub Graph:

A graph whose vertices and edges are subset of another graph it called sub graph.

$G = (V, E)$ and Subgraph $G' = (V', E')$.



$V = \{1, 2, 3, 4\}$



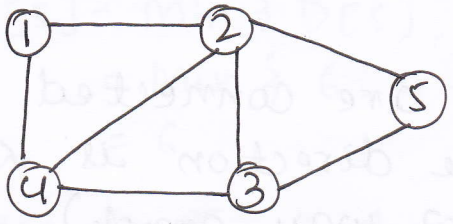
(Sub-graph).

$V' = \{1, 2, 3\}$

$E' = \{e_1, e_2\}$.

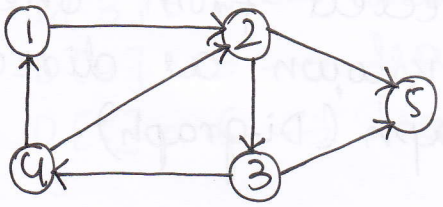
4. Adjacent and Incident Vertices.

Vertices i and j are adjacent vertices iff (i, j) is an edge in the graph.
The edge (i, j) is incident on the vertices i and j.



Adjacent of Vertex ① = vertex ② & ④.
Adjacent of vertex ② = vertex ①, ④, ③, ⑤
Adjacent of vertex ③ = Vertex ②, ④, ⑤.

Incident of vertex ① = 2 Edges.
Incident of vertex ④ = 3 Edges.
Incident of Vertex ② = 4 Edges.



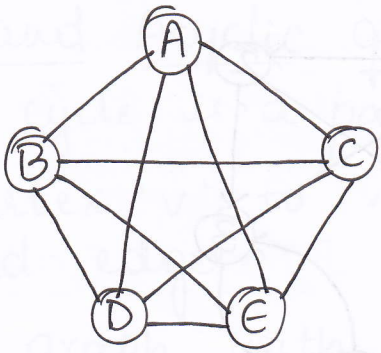
Adjacent of vertex ① = ②.
Adjacent of vertex ② = ③ & ⑤
Adjacent of vertex ③ = ④

Incident of vertex ① = 1 Edge
Incident of vertex ④ = 1 Edge
Incident of vertex ② = 2 Edge
Incident of vertex ⑤ = 2 Edge

5. Complete Graph

A complete Graph is an undirected graph with an edge between each pair of vertices.

The complete Graph with n vertices having $n(n-1)/2$ Edges.



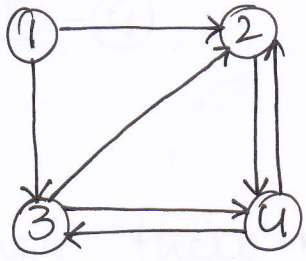
if $n = 5$

$5(5-1)/2 = 10$ Edges.

6. Indegree and Outdegree.

The no. of Edges are incident to the particular Vertex is known as Indegree.

The no. of Edges are directed to the other Vertices is known as Outdegree.



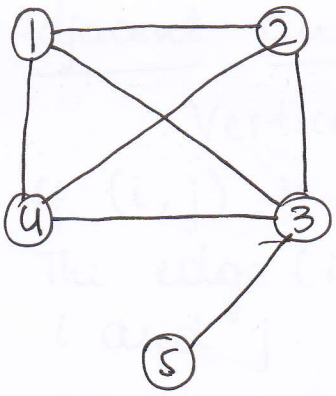
Indegree of ① = 0
Indegree of ② = 3
Indegree of ③ = 2
Indegree of ④ = 2

Outdegree of ① = 2
Outdegree of ② = 1
Outdegree of ③ = 2
Outdegree of ④ = 2

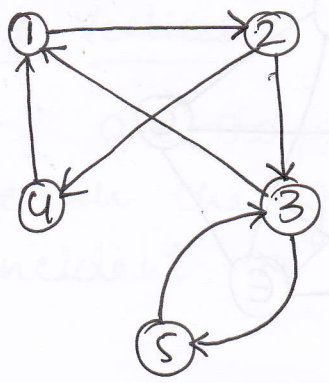
7. Connected and Strongly connected graph.

An undirected graph is called connected Graph, for every pair of vertices there exists a path b/w them.

For a directed graph, for every pair of vertices (i, j) there exists a path from i to j and a path from j to i, then we say that the graph is strongly connected graph.



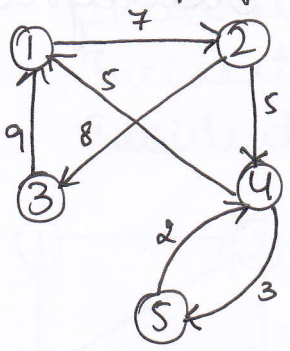
Connected Graph.



Strongly connected graph

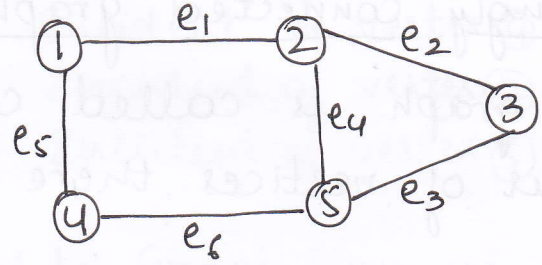
8. Weighted Graph.

A directed (or) undirected Graph assigns weights to every edge is called weighted Graph.



9. Path:

A path is a trip beginning at the same vertex and passing thro' several vertices and back to the same vertex.



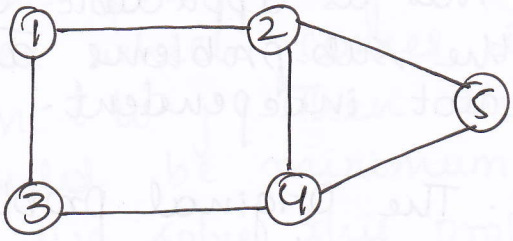
① e_1 ② e_2 ③ e_3 ⑤ e_6 ④ e_5 ①

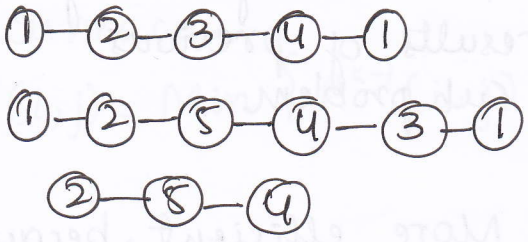
V_0 V_0.

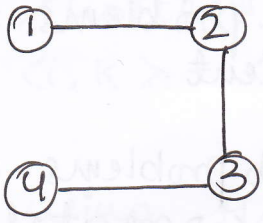
10. Cyclic and Acyclic graph.

A cycle is a path of non-zero length from vertex 'v' to same vertex 'v' with no repeated edges.

A graph with no cycle is called acyclic graph.



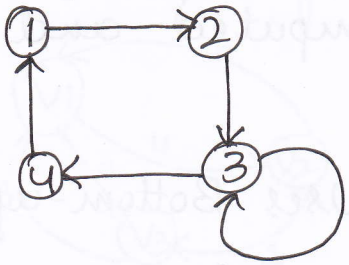




Acyclic

11. Loop

When there is an edge (i, i) that is connecting vertices themselves is called loop.



Dynamic Programming.

is a method of solving the problem with overlapping sub-problems. This method works by dividing the problem into sub problems. and the solution can be obtained by using the results of the previous sub problem.

Once a sub problem is solved, the result is stored in a table and never recalculated,

another sub-problem.

The word programming is the process of obtaining the plan to get the optimal solution.

Difference b/w D & C and Dynamic programming.

Divide and Conquer	Dynamic Programming.
1. This is applicable when the sub-problems are independent.	1. This is applicable when the sub-problems are not independent.
2. The sub problems are solved separately and combined to get the solution for the original problem.	2. The Original problem is solved by using the results of previous sub problem.
3. Not efficient because of re-computations.	3. More efficient because re-computations are not done.
4. Every instance of the sub problem is re-computed and is not stored.	4. Only one instance of the sub problem is computed and stored.
5. Uses Top-down Approach.	5. Uses Bottom-up approach.

Similarity b/w D & C and Dynamic programming.

Both Techniques solve the problem by dividing the problem into subproblems. Using the solutions of sub-problems, the solution for the larger instance of the problem can be obtained.

All pairs Shortest Path.

In a given connected, directed, and weighted graph 'G', the objective of the problem is to find the shortest path from every vertex 'i' to every vertex 'j' in graph 'G'.

We can apply dynamic programming principle to this problem by considering 'K' being a vertex which comes in the shortest path from i to j. Then the path $\langle i, K \rangle + \langle K, j \rangle$ should be minimum.

We solve this problem by using the recurrence,

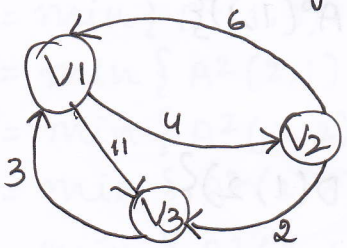
$$A^K(i,j) = \text{Min}\{A^{K-1}(i,j), A^{K-1}(i,K) + A^{K-1}(K,j)\}$$

where $K > 0$

$K = 1, 2, 3, 4, - - -$

$A^K(i,j)$ represents the length of the shortest path from the vertex i to vertex j.

1. Find the all pairs shortest path for the given directed graph.



Soln: WKT $A^K(i,j) = \text{Min}\{A^{K-1}(i,j), A^{K-1}(i,K) + A^K(K,j)\}$

The cost matrix for the given graph.

A^0	V_1	V_2	V_3
V_1	0	4	11
V_2	6	0	2
V_3	3	∞	0

Step 1: K=1, ie., going from i to j through v_1

$A^1(1,1) = \text{Min}\{A^0(1,1), A^0(1,1) + A^0(1,1)\}$
$= \text{Min}\{0, 0+0\}$
$= 0$

$A^1(1,2) = \text{Min}\{A^0(1,2), A^0(1,1) + A^0(1,2)\}$
$= \text{Min}\{4, 0+4\}$
$= 4$

$A^1(1,3) = \text{Min}\{A^0(1,3), A^0(1,1) + A^0(1,3)\}$
$= \text{Min}\{11, 0+11\}$
$= 11$

$A^1(2,1) = \text{Min}\{A^0(2,1), A^0(2,1) + A^0(1,1)\}$
$= \text{Min}\{6, (6+0)\}$
$= 6$

$A^1(2,2) = \text{Min}\{A^0(2,2), A^0(2,1) + A^0(1,2)\}$
$= \text{Min}\{0, 6+4\}$
$= 0$

$A^1(2,3) = \text{Min}\{A^0(2,3), A^0(2,1) + A^0(1,3)\}$
$= \text{Min}\{2, 6+11\}$
$= 2$

$A^1(3,1) = \text{Min}\{A^0(3,1), A^0(3,1) + A^0(1,1)\}$
$= \text{Min}\{3, 3+0\}$
$= 3$

$A^1(3,2) = \text{Min}\{A^0(3,2), A^0(3,1) + D(1,2)\}$
$= \text{Min}\{\infty, 3+4\}$
$= 7$

$A^1(3,3) = \text{Min}\{A^0(3,3), A^0(3,1) + A^0(1,3)\}$
$= \text{Min}\{0, 3+11\}$
$= 0$

A^1	V_1	V_2	V_3
V_1	0	4	11
V_2	6	0	2
V_3	3	7	0

$= A^1(i,j)$

Step2: K = 2

$A^2(1,1) = \text{Min}\{A^1(1,1), A^1(1,2) + A^1(2,1)\} = \text{Min}\{0, 0\} = 0$

$A^2(1,2) = \text{Min}\{A^1(1,2), A^1(1,2) + A^1(2,2)\} = \text{Min}\{4, 4+0\} = 4$

$A^2(1,3) = \text{Min}\{A^1(1,3), A^1(1,2) + A^1(2,3)\} = \text{Min}\{11, 4+2\} = 6$

$A^2(2,1) = \text{Min}\{A^1(2,1), A^1(2,2) + A^1(2,1)\} = \text{Min}\{6, 0+6\} = 6$

$A^2(2,2) = \text{Min}\{A^1(2,2), A^1(2,2) + A^1(2,2)\} = \text{Min}\{0, 0\} = 0$

$A^2(2,3) = \text{Min}\{A^1(2,3), A^1(2,2) + A^1(2,3)\} = \text{Min}\{2, 0+2\} = 2$

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

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

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

$A^2(i,j) =$

A^2	V_1	V_2	V_3
V_1	0	4	6
V_2	6	0	2
V_3	3	7	0

Step3: K = 3

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

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

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

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

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

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

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

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

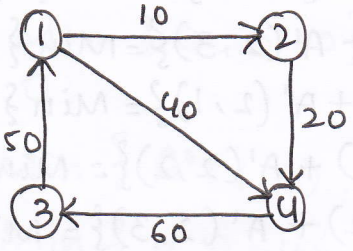
$A^3(3,3) = \text{min}\{A^2(3,3), A^2(3,3) + D^2(3,3)\} = \text{Min}\{0, 0+0\} = 0$

$A^3(i,j) =$

D^3	V_1	V_2	V_3
V_1	0	4	6
V_2	5	0	2
V_3	3	7	0

The adjacency matrix $A^3(i,j)$ gives all pairs

2. Find the all pairs shortest path for the given directed graph.



Algorithm for all pairs Shortest path.

cost[1:n, 1:n] is the cost adjacency matrix of a graph with n vertices.

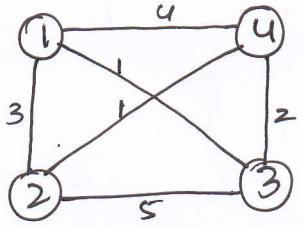
A [i, j] is the shortest path from vertex 'i' to vertex 'j'.

```
Alg. AllpairsShortestpath (cost, A, n)
{
    for i=1 to n do
      for j=1 to n do
      A[i, j] = cost[i, j];
  for (K=1 to n) do
     for i=1 to n do
        for j=1 to n do
     A[i, j] = min ( A[i, j] , A[i, K] + A[K, j]);
   End for
  End for
 End for
}.
```

Travelling Salesman Problem.

In this problem, A Salesman must visit 'n' cities, visiting each city exactly once and finishing at the city he started from. There is an integer cost c(i, j) to travel from city i to city j. The Salesman wishes to make the tour whose total cost is minimum.

Eg:



In the above graph, let us assume ① be the source vertex.

The min. cost tour = ① — ③ — ④ — ② — ①

∴ The min. cost = 7.

let G(V, E) be a graph with edge costs c(i, j). let 'V' be as set of cities, a cost of the tour is the sum of the cost of the edges on the tour. The Travelling Salesman problem is to find a tour of min. cost.

The tour will start from a source vertex 'V', visits the remaining vertices exactly once and terminates at the starting vertex.

let 'K' be some vertex in V-1 and path from vertex K to vertex 1 must be a shortest path going through all vertices in $V - \{1, K\}$ exactly once.

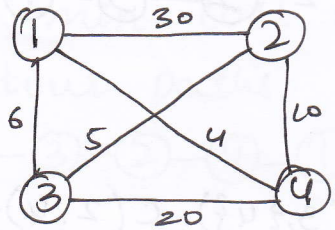
General eqn for solving this problem is

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

let $g(i,S)$ be the length of a shortest path starting vertex i.

Problems

1. Find the optimal tour cost for the given undirected graph. connected graph ?



Soln

Create the cost matrix for the given graph.

C(i,j)	1	2	3	4
1	0	30	6	4
2	30	0	5	10
3	6	5	0	20
4	4	10	20	0

Step1: for $\{S\} = 0$

Starting from the node ①

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

$g(2,0) = c(2,1) = 30$

$g(3,0) = c(3,1) = 6$

$g(4,0) = c(4,1) = 4$

Step 2: for $\{s\} = 1$

Coming to node ①, going thro' all the nodes in $|s|$

$g(2, \{3\}) = \min\{c(2,3) + g(3,0)\} = 5 + 6 = 11$

$g(2, \{4\}) = \min\{c(2,4) + g(4,0)\} = 10 + 4 = 14$

$g(3, \{2\}) = \min\{c(3,2) + g(2,0)\} = 5 + 30 = 35$

$g(3, \{4\}) = \min\{c(3,4) + g(4,0)\} = 20 + 4 = 24$

$g(4, \{2\}) = \min\{c(4,2) + g(2,0)\} = 10 + 30 = 40$

$g(4, \{3\}) = \min\{c(4,3) + g(3,0)\} = 20 + 6 = 26$

Minimum = 11 & path = ② — ③ — ①.

Step 3: for $|s| = 2$

$g(2, \{3,4\}) = \min\{c(2,3) + g(3,\{4\}), c(2,4) + g(4,\{3\})\}$

$= \min\{5 + 24, 10 + 26\} = \min\{29, 36\}$.

$= 29$ is min. ② — ③ — ④ — ①

$g(3, \{2,4\}) = \min\{c(3,2) + g(2,\{4\}), c(3,4) + g(4,\{2\})\}$

$= \min\{5 + 14, 20 + 40\}$

$= \min\{19, 60\}$

$= 19$ is min. ③ — ② — ④ — ①

$g(4, \{2,3\}) = \min\{c(4,2) + g(2,\{3\}), c(4,3) + g(3,\{2\})\}$

$= \min\{10 + 11, 20 + 35\}$

$= \min\{21, 55\}$

$= 21$ is min. ④ — ② — ③ — ①.

∴ The minimum cost = 19 & the path =

③ — ② — ④ — ①.

Step 4: $|s| = 3$

$g(1,\{2,3,4\}) = \min\{c(1,2)+g(2,\{3,4\}),$
$c(1,3)+g(3,\{2,4\}),$
$c(1,4)+g(4,\{2,3\})\}$

$= \min\{30+29, 6+19, 4+21\}$

$= \min\{59, 25, 25\}.$

path of 59 = ①—②—③—④—①

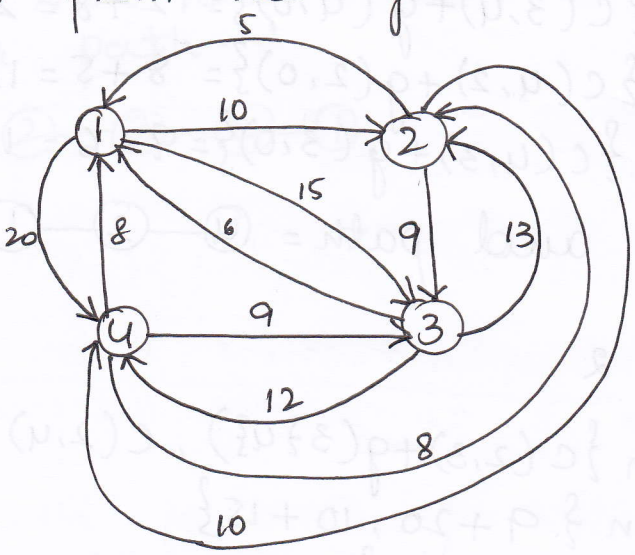
path of 25 = ①—③—②—④—①

path of 25 = ①—④—②—③—①

∴ The min. cost = 25 and there exists 2 tour paths.

①—③—②—④—① and ①—④—②—③—①.

2. Find the min. tour cost using dynamic programming technique starting from the vertex ① from the given below directed graph.



C(i,j)	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

Step 1: for $|s| = 0$

Starting from the node ①

$$g(i, s) = \min_{j \in s} \{ C(i,j) + g(j, s - \{j\}) \}.$$

$g(2, \phi) = C(2,1) = 5$

$g(3,0) = C(3,1) = 6$

$g(4,0) = C(4,1) = 8$

Minimum = 5 & path = ② — ①

Step 2: for $\{s\} = 1$

Coming to node ①. going thro' all the nodes in $|s|$.

$g(2, \{3\}) = \min \{ C(2,3) + g(3,0) \} = 9 + 6 = 15$

$g(2, \{4\}) = \min \{ C(2,4) + g(4,0) \} = 10 + 8 = 18$

$g(3, \{2\}) = \min \{ C(3,2) + g(2,0) \} = 13 + 5 = 18$

$g(3, \{4\}) = \min \{ C(3,4) + g(4,0) \} = 12 + 8 = 20$

$g(4, \{2\}) = \min \{ C(4,2) + g(2,0) \} = 8 + 5 = 13$

$g(4, \{3\}) = \min \{ C(4,3) + g(3,0) \} = 9 + 6 = 15$

Minimum = 13 and path = ④ — ② — ①

Step 3: for $|s| = 2$

$g(2, \{3,4\}) = \min \{ C(2,3) + g(3, \{4\}), C(2,4) + g(4, \{3\}) \}$

$= \min \{ 9 + 20, 10 + 15 \}$

$= \min \{ 29, 25 \}$

$= 25$ is min. ② — ④ — ③ — ①.

$g(3, \{2,4\}) = \min \{ C(3,2) + g(2, \{4\}), C(3,4) + g(4, \{2\}) \}$

$= \min \{ 13 + 18, 12 + 13 \}$

$= \min \{ 31, 25 \}$

$= 25$ is min. ③ — ④ — ② — ①.

$g(4, \{2,3\}) = \text{Min}\{ c(4,2) + g(2,\{3\}),\ c(4,3) + g(3,\{2\}) \}$

$= \text{Min}\{ 8 + 15,\ 9 + 18 \}$

$= \text{Min}\{ 23,\ 27 \}$

$= 23$ is min. ④—②—③—①.

∴ The minimum cost = 23 & the path = ④—②—③—①.

Step 4: $|S| = 3$.

$g(1, \{2,3,4\}) = \text{Min}\{ c(1,2) + g(2,\{3,4\}),$
$c(1,3) + g(3,\{2,4\}),$
$c(1,4) + g(4,\{2,3\}) \}.$

$= \text{Min}\{ 10+25,\ 15+25,\ 20+23 \}$

$= \text{Min}\{ 35,\ 40,\ 43 \}.$

path of 35 = ①—②—③—④—①
path of 40 = ①—③—②—④—①
path of 43 = ①—④—②—③—①

∴ The min. cost = 35 and there exists a path.

①—②—③—④—①.

Flow Shop Scheduling.

Flow Shop Scheduling problem is one of the most imp. problem in the area of production management. There are 'n' jobs, each requiring 'm' tasks.

$T_{1i}, T_{2i}, T_{3i}, \ldots\ldots T_{mi}$ $\quad 1 \leq i \leq n$. to be performed.

Task T_{ji} is to be performed on processor 'P_j'. $\quad 1 \leq j \leq m$.

The time required to complete the task T_{ji} is t_{ji}.

The main objective is to find the job ~~Sch~~ sequence on the processors ~~which minimizes~~ to make max. of the completion time of all operations.

Constraints

1. A schedule for the 'n' jobs is an assignment of tasks to time intervals on the processors.
2. Task T_{ji} must be assigned to processor P_j
3. No processor may have more than one task assigned to it in any time interval.
4. For any job i, the processing of task T_{ji} cannot started until task T_{j-1i} has been completed.

There are 2 types of Schedule.

1. Non-Preemptive Schedule - The processing of a task on any ~~processing~~ processor is not terminated until the task is completed.

2. Preemptive Schedule — The processing of a task on any processor is terminated before completing the task is called preemptive schedule.

The finishing time $F_i(S)$ of job 'i' is the time at which all tasks of job 'i' have been completed in schedule 'S'.

The finishing time $F(S) = \max_{1 \le i \le n} \{ f_i(S) \}$

The mean flow time, MFT(S) is defined to be

$$MFT(S) = \frac{1}{n} \sum_{1 \le i \le n} f_i(S)$$

problem

Let n=2 jobs to be scheduled on 3-processor $\{P_1, P_2, P_3\}$.

The task time are given $J = \begin{bmatrix} 2 & 0 \\ 3 & 3 \\ 5 & 2 \end{bmatrix}$

Soln

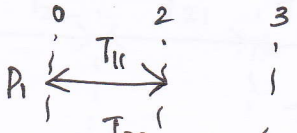
X. Non-preemptive Schedule.

The matrix representation for the given task is

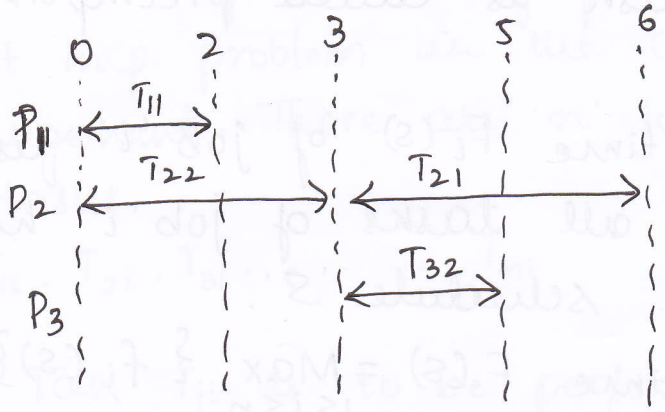
$$J = \begin{matrix} \\ 1 \\ 2 \\ 3 \\ j \end{matrix} \begin{matrix} \overset{1}{\longrightarrow} & \overset{2}{i} \end{matrix} \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ T_{31} & T_{32} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 3 & 3 \\ 5 & 2 \end{bmatrix}$$

Here $T_{11} = 2$, $T_{12} = 0$, $T_{21} = 3$, $T_{22} = 3$, $T_{31} = 5$, $T_{32} = 2$

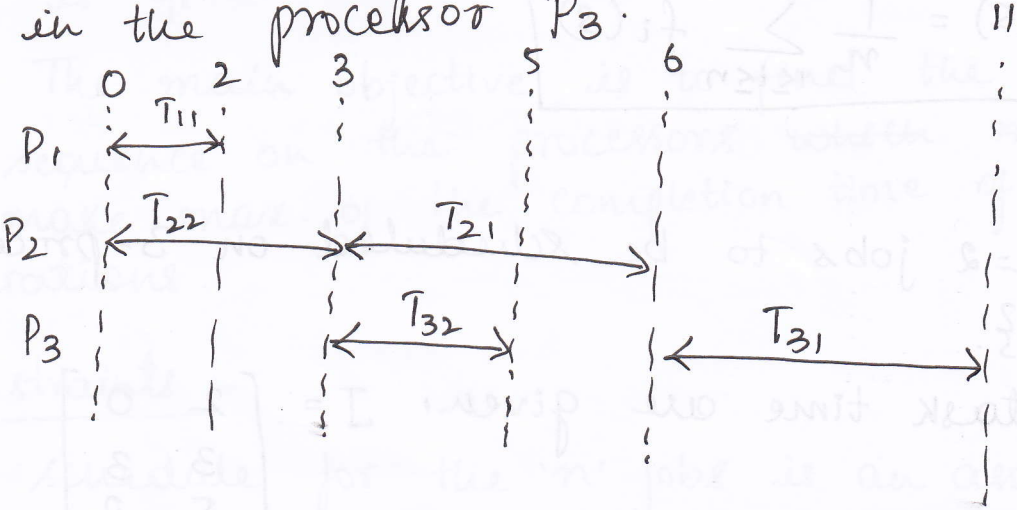
Step1: Assign T_{11} to P_1 & T_{22} to P_2 at Time 0.



Step 2: Since T_{11} & T_{22} is completed.
let T_{21} and T_{31} can start execution.



Step 3: Since T_{21} and T_{32} are completed and the remaining task has been executed in the processor P_3.



All the tasks are completed successfully

$\therefore$ Finishing time $f_1(S) = 11$ ($\because T_{31}$ is completed at time = 11).

Finishing time $F_2(S) = 5$ ($\because T_{32}$ is completed at time = 5).

Finishing Time $F(S) = \underset{1 \le i \le n}{\text{Max}} \{ F_i(S) \}$

$= \text{Max} \{ F_1(S), F_2(S) \}$

$= \text{Max} \{ 11, 5 \}$

$F(S) = 11.$

Mean Time flow.

$$MTF(S) = \frac{1}{n} \sum_{1 \le i \le n} f_i(S).$$

$$= \frac{1}{n}\{F_1(S) + F_2(S)\}$$

$$= \frac{1}{2}\{11 + 5\}$$

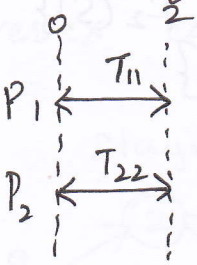
$$= \frac{16}{2}$$

$$MTF(S) = 8$$

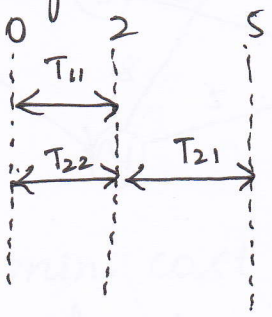
Preemptive Schedule.

$$J = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ T_{31} & T_{32} \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 3 & 3 \\ 5 & 2 \end{bmatrix}$$

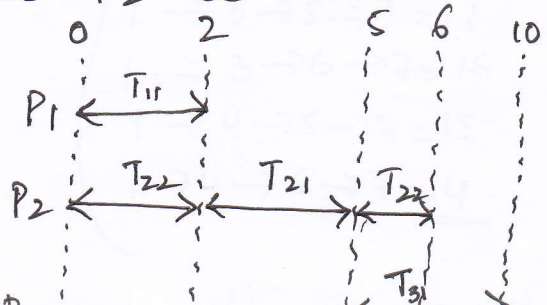
Step 1: Assigning T_{11} & T_{22} to P_1 & P_2 resp. at time zero.



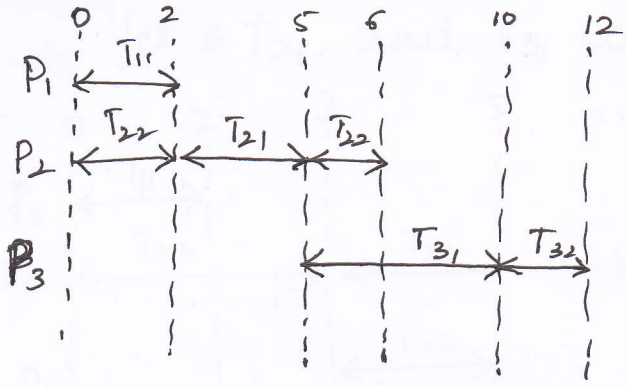
Step 2: Assign T_{21} to P_2 at time = 2.



Step 3: Assign T_{31} to P_3 as T_{21} is already completed but T_{22} is not completed yet, then assign T_{22} to P_2 at time = 5.



Step 4: Assign T_{32} to P_3 at time = 10.



All the tasks are completed successfully.

∴ Finishing time $F_1(S) = 10$ ($\because T_{31}$ is completed at time = 10)

Finishing time $F_2(S) = 12$ ($\because T_{32}$ is completed at time = 12).

Finishing Time $F(S) = \max_{1 \le i \le n} \{ f_i(S) \}$

$= \text{Max} \{ F_1(S), F_2(S) \}$

$= \text{Max} \{ 10, 12 \}$

$f(S) = 12$

Mean flow Time

$$MFT(S) = \frac{1}{n} \sum_{1 \le i \le n} F_i(S)$$

$$= \frac{1}{n} (F_1(S) + F_2(S))$$

$$= \frac{1}{2} (10 + 12) = \frac{22}{2}$$

$MFT(S) = 11$

Multistage Graph.

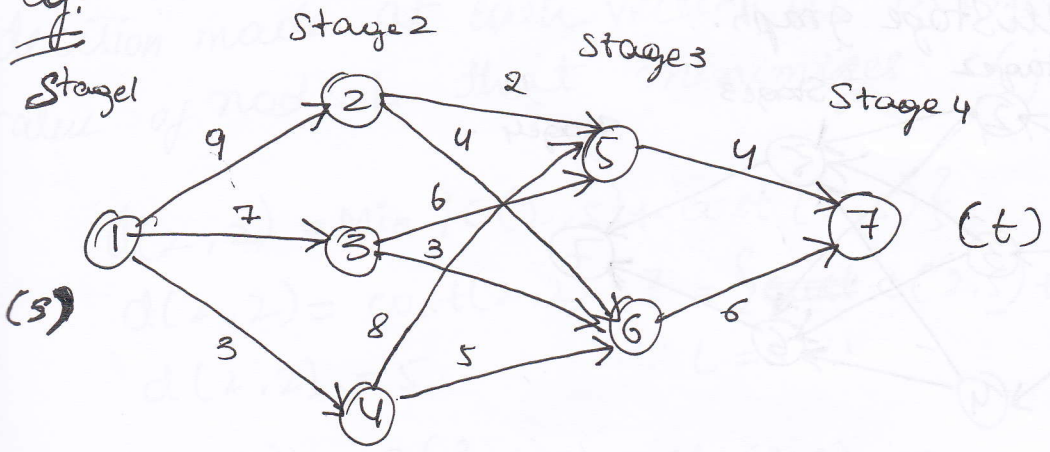
A multistage graph problem is to find the min. cost path from the source vertex 'S' to the Destination vertex (or) Terminal node (T) using Dynamic programming.

A multistage graph $G = (V, E)$ is a directed graph. In which the vertices are partitioned into $K >= 2$ disjoint sets say $V_1, V_2, V_3, \ldots V_K$. If (u, v) is an edge in E then u belongs to V_K. and v belongs to V_{i+1} where $1 \leq i < K$.

The no. of vertices in each of the first and last sets V_i and V_K is 1.

Let the node in the first set V_1 be called Source (s) and node in the last set V_K be called Terminal (T). Let $C(u, v)$ be the cost of traversing from vertex u to vertex v.

Eg:



Find the min. cost to travel from source vertex ① to terminal vertex ⑦.

$$= \min \begin{cases} 1 \to 2 \to 5 \to 7 = 15 \\ 1 \to 2 \to 6 \to 7 = 19 \\ 1 \to 3 \to 5 \to 7 = 17 \\ 1 \to 3 \to 6 \to 7 = 16 \\ 1 \to 4 \to 5 \to 7 = 15 \\ 1 \to 4 \to 6 \to 7 = 14. \end{cases}$$

∴ The min. cost to travel from source s to

i.e., 1 → 4 → 6 → 7

Dynamic programming solution to multistage graph.

1. Forward approach.

A DP soln. for a K stage graph is obtained by first noticing that every 's' to 't' path is the result of the sequence of K-2 decisions.

let cost of i, j be the cost of this path. Therefore, the

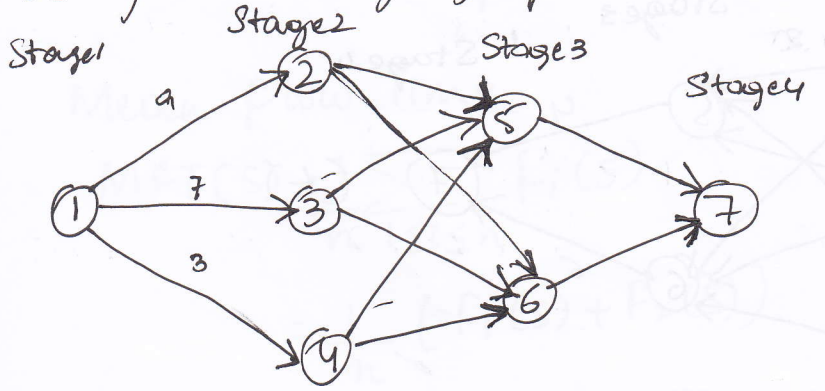
$$cost(i,j) = Min\{c(j,L) + cost(i+1, L)\}.$$

where L belongs to v_{i+1} & $(j,L) \in E$.

we go from vertex j in stage i to the vertex L in stage i+1 which has a min. cost reaching to the destination indicated by cost(i+1, L) as per the principle of Optimality.

problem 1:-

Find the min. cost path from the node ① to ⑦ using multistage graph.



Soln: we have 4 stages and hence the value of K is 4. First find out the cost from (K-1)th stage to K stage.

Step1: $cost(i,j) = Min\{c(j,L) + cost(i+1, L)\}$.

Stage 3 → cost(3,5) = 5
cost(3,6) = 6

Step 2: calculating min. cost from stage2.

$cost(2,2) = Min\{c(2,5) + cost(3,5), c(2,6) + cost(3,6)\}$

$= Min\{2+5, 4+6\}$

$cost(2,3) = Min\{c(3,5) + cost(3,5),\ c(3,6) + cost(3,6)\}$

$= Min\{6+5, 3+6\}$

$= Min\{11, 9\} = 9$

$cost(2,4) = Min\{c(4,5) + cost(3,5),\ c(4,6) + cost(3,6)\}$

$= Min\{8+5, 5+6\}$

$= Min\{13, 11\} = 11.$

Step 3:

$cost(1,1) = Min\{c(1,2) + cost(2,2),\ c(1,3) + cost(2,3),$

$c(1,4) + cost(2,4)\}$

$= Min\{9+7,\ 7+9,\ 3+11\}$

$= 14.$

Find the determined path

The path can be determined if we record the decision made at each vertex. Let $D(i,j)$ be the value of node L that minimizes $c(j,L) + cost(i+1,L)$

$d(2,2) = Min\{c(2,5) + cost(3,5)\}$

$d(2,2) = cost(2,2) = 7 = \{cost\ c(2,5) + cost(3,5)\}$

$d(2,2) = 5 \quad \because L = 5.$

$d(2,3) = c(3,6) + cost(3,6) = 9$

$d(2,3) = 6.$

$d(2,4) = c(4,6) + cost(3,6) = 11$

$d(2,4) = 6$

$d(1,1) = c(1,4) + cost(2,4) = 3$

$d(1,1) = 4.$

let the min. cost path be $S=1, v_2, v_3 \dots v_{k-1}, t$.

Here $S=1$

$v_2 = d(1,1) = 4$

$v_3 = d(2, d(1,1)) = d(2,4) = 6$.

$t = 7$.

$1 \to v_2 \to v_3 \to 7$

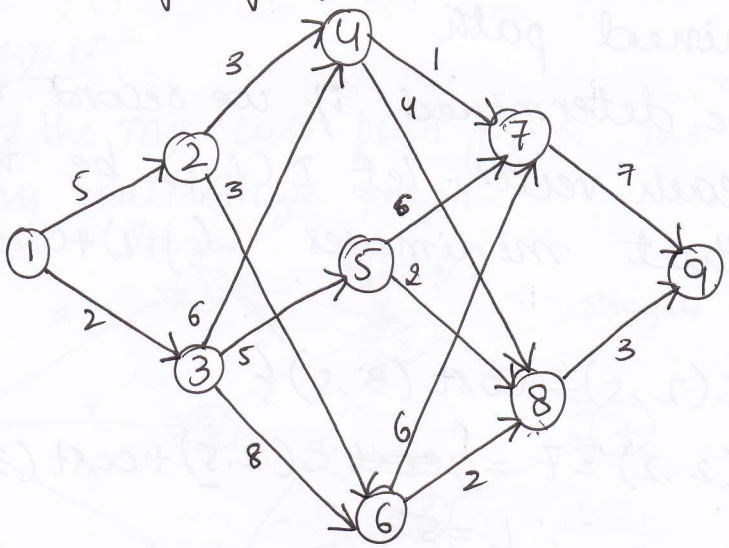
$1 \to 4 \to 6 \to 7$.

$\therefore$ path is ① → ④ → ⑥ → ⑦

is the min. cost path.

problem 3:

Find the min. cost path from starting ① to ⑨ using multistage graph.



Stage1 Stage2 Stage3 Stage4 Stage5.

Step1: Here there are 5-stages $\therefore k=5$

let us start from $(k-1)^{th}$ stage i.e., 4^{th} stage.

cost (4,7) = 7

cost (4,8) = 3

Step2: calculating minimum cost from stage 3.

cost (3,4) = Min { c(4,7) + cost(4,7), c(4,8) + cost(4,8) }

= Min (1+7, 4+3) = Min {8, 7}

$cost(3,5) = Min\{c(5,7) + cost(4,7), c(5,8) + cost(4,8)\}$.

$= Min\{6+7, 2+3\} = Min\{13, \underline{5}\}$

$d(3,5) = 8$.

$cost(3,6) = Min\{c(6,7) + cost(4,7), c(6,8) + cost(4,8)\}$

$= Min\{6+7, 2+3\} = Min\{13, \underline{5}\}$

$d(3,6) = 8$

Step 3:

$cost(2,2) = Min\{c(2,4) + cost(3,4), c(2,6) + cost(3,6)\}$

$= Min\{3+7, 3+5\} = Min\{10, \underline{8}\}$

$d(2,2) = 6$

$cost(2,3) = Min\{c(3,4) + cost(3,4), c(3,5) + cost(3,5), c(3,6) + cost(3,6)\}$

$= Min\{6+7, 5+5, 8+5\}$

$= Min\{13, \underline{10}, 13\}$.

$d(2,3) = 5$

Step 4:

$cost(1,1) = Min\{c(1,2) + cost(2,2), c(1,3) + cost(2,3)\}$

$= Min\{5+8, 2+10\}$

$= Min\{13, \underline{12}\}$

$d(1,1) = 3$.

calculating the path s, v_2, v_3, v_4, t.

$s = 1$

$v_2 = d(1,1) = 3$

$v_3 = d(2, v_2) = d(2,3) = 5$

$v_4 = d(3, v_3) = d(3,5) = 8$

$t = 9$.

∴ The min. cost path =

$1 \longrightarrow 3 \longrightarrow 5 \longrightarrow 8 \longrightarrow 9$.

Algorithm for Multistage graph using forward Approach.

Let $G = (V, E)$ is a directed k stage weighted graph with n vertices. So E is set of edges.
$c(i,j)$ is the cost of (i,j) edge.
$P[1:k]$ is a min. cost path.

```
Algorithm. fgraph ( G, k, n, p).
Step 1: cost[n] = 0; // cost of terminal vertex is zero.
Step 2: To compute the cost of[ j] from k-1 stage
          to 1st stage.
        for ( j = n-1 to 1 do
        {
          let L be a vertex such that
          (j, L) is an edge of G.
           c [j, L] + cost [L] is minimum.
          cost [j] = c [j, L] + cost [L];
           d[j] = L;
        }
Step 3: To find min. cost path.
          p[1] = 1
```

```
for ( j = 2 to k-1 . do
    p[j] = d ( p[j-1]);
```

Step 4: Exit.

Backward Approach.

The multistage graph problem can also be solved using backward approach.

let bp(i, j) be the min. cost path from vertex s to vertex j in V_i.

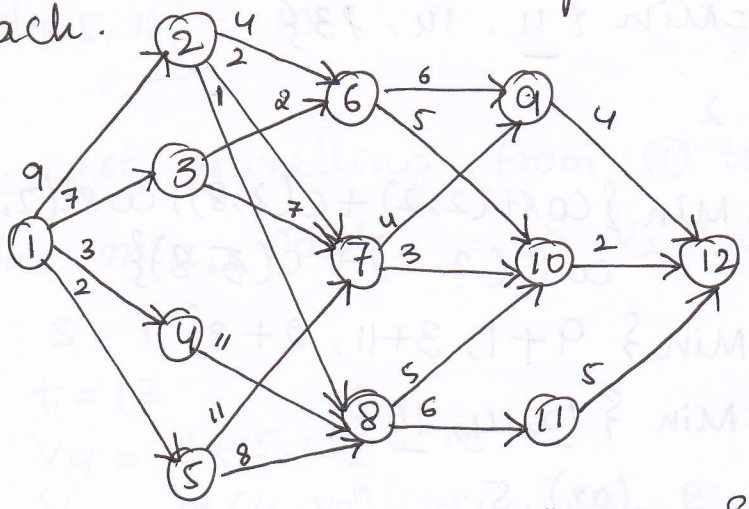
let bcost (i,j) be the cost of bp(i,j)

i.e., bcost (i,j) = Min{ bcost (i-1, L) + c (L, j)}

$L \in V_{i-1}$ $(L, j) \in E$.

Problem:

Find the min. cost path from the starting node 1 to node 12 using multistage backward approach.



Stage 1 Stage 2 Stage 3 Stage 4 Stage 5.

Step 1: calculating from stage 1.

cost (1, 1) = 0

Step 2: calculating Min. cost from stage 2.

cost (2, 2) = cost(1, 1) + c(1, 2) = 0 + 9 = 9

$cost(2,3) = cost(1,1) + c(1,3) = 0 + 7 = 7$

$d(2,3) = 1$

$cost(2,4) = cost(1,1) + c(1,4) = 0 + 3 = 3$

$d(2,4) = 1$

$cost(2,5) = cost(1,1) + c(1,5) = 0 + 2 = 2.$

$d(2,5) = 1$

Step 3: calculating min. cost from stage 3.

$cost(3,6) = Min\{cost(2,2) + c(2,6),$
$cost(2,3) + c(3,6)\}$
$= Min\{9+4, 7+2\} = Min\{13, 9\}$
$= 9.$

$d(3,6) = 3$

$cost(3,7) = Min\{cost(2,2) + c(2,7), cost(2,3) +$
$c(3,7), cost(2,5) + c(5,7)\}$
$= Min\{9+2, 7+7, 2+11\}$
$= Min\{11, 14, 13\}$

$d(3,7) = 2$

$cost(3,8) = Min\{cost(2,2) + c(2,8), cost(2,4) + c(4,8),$
$cost(2,5) + c(5,8)\}$
$= Min\{9+1, 3+11, 2+8\}$
$= Min\{10, 14, 10\}$

$d(3,8) = 2$ (or) $5.$

Step 4: calculating min. cost from stage 4.

$cost(4,9) = Min\{cost(3,6) + c(6,9),$
$cost(3,7) + c(7,9)\}$
$= Min\{9+6, 11+4\}$
$= Min\{15, 15\}$

$$cost(4,10) = Min\{cost(3,6)+c(6,10),\ cost(3,7)+c(7,10),\ cost(3,8)+c(8,10)\}$$
$$= Min\{9+5,\ 11+3,\ 10+5\}$$
$$= Min\{14,14,15\}$$

$d(4,10) = 6 / 7$.

$$cost(4,11) = cost(3,8)+c(8,11)$$
$$= 10+6 = 16.$$

$d(4,11) = 8$.

Step 5:

$$cost(5,12) = Min\{cost(4,9)+c(9,12),\ cost(4,10)+c(10,12),\ cost(4,11)+c(11,12)\}$$
$$= Min\{15+4,\ 14+2,\ 16+5\}$$
$$= Min\{19,\ 16,\ 21\} = 16$$

$d(5,12) = 10$.

Min. cost travelling from ① to terminal vertex ⑫

∴ The min. path = $S\ V_2\ V_3\ V_4\ t$.

$s = 1$

$t = 12$

$V_4 = d(5,12) = 10$

$V_3 = d(4, V_4) = d(4,10) = 6$ (or) 7

$V_2 = d(3, V_3) = d(3,6)$ (or) $d(3,7) = 3 / 2$.

∴ path = ① → ③ → ⑥ → ⑩ → ⑫. = 16.

= ① → ② → ⑦ → ⑩ → ⑫. = 16.

Algorithm for multistage graph using backward approach.

$G=(V, E)$ is a directed k stage weighted graph with n vertices. So E is a set of edges and $C(i,j)$ is the cost of (i,j) edge.

$p[1:k]$ is the min. cost path.

```
Alg. Bgraph(G, k, n, p)
Step1: bcost[1] = 0;
Step2: To compute the cost[j] from stage 1st to
       stage k.
       for j=2 to n do.
       {
         Let L be a vertex such that (L, j)
              is an edge of G.
         C[j,L] + cost[L] is min;
         cost[j] = cost[L] + C[j,L];
         d[j] = L;
       }
Step3: To find min. cost path
         P[1] = 1
         P[k] = n
         for j=2 to k-1 do
           p[j] = d[p[j+1]]
Step 4: Exit.
```