UNIT-3

DYNAMIC PROGRAMMING

General method-multistage graphs-all pair shortest path algorithm-0/1 knapsack and traveling salesman problem-chained matrix multiplication-approaches using recursion-memory functions

BASIC SEARCH AND TRAVERSAL TECHNIQUES

The techniques-and/or graphs-bi_connected components-depth first search-topological sorting-breadth first search.

DYNAMIC PROGRAMING

- The idea of dynamic programming is thus quit simple: avoid calculating the same thing twice, usually by keeping a table of known result that fills up a sub instances are solved.
- 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.
- 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 and so will not be generated.

ALL PAIR SHORTEST PATH

- Let G=<N,A> be a directed graph 'N' is a set of nodes and 'A' is the set of edges.
- Each edge has an associated non-negative length.
- We want to calculate the length of the shortest path between each pair of nodes.

- Suppose the nodes of G are numbered from 1 to n, so N={1,2,...N},and suppose G matrix L gives the length of each edge, with L(i,j)=0 for i=1,2...n,L(i,j)>=for all i & j, and L(i,j)=infinity, if the edge (i,j) does not exist.

- The principle of optimality applies: if k is the node on the shortest path from i to j then the part of the path from i to k and the part from k to j must also be optimal, that is shorter.

- First, create a cost adjacency matrix for the given graph.

- Copy the above matrix-to-matrix D, which will give the direct distance between nodes.

- We have to perform N iteration after iteration k.the matrix D will give you the distance between nodes with only (1,2...,k)as intermediate nodes.

- At the iteration k, we have to check for each pair of nodes (i,j) whether or not there exists a path from i to j passing through node k.

COST ADJACENCY MATRIX:

$$D0 = L = \begin{vmatrix} 0 & 5 & \propto & \propto \\ 50 & 0 & 15 & 5 \\ 30 & \propto & 0 & 15 \\ 15 & \propto & 5 & 0 \end{vmatrix}$$

$$\begin{array}{c|} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{vmatrix} 7 & 5 & \propto & \propto \\ 7 & \propto & \propto & 2 \\ \propto & 3 & \propto & \propto \\ 4 & \propto & 1 & \propto \end{vmatrix} \quad \begin{vmatrix} 11 & 12 & - & - \\ 21 & - & - & 24 \\ - & 32 & - & - \\ 41 & - & 43 & - \end{vmatrix}$$

vertex 1:

$$\begin{vmatrix} 7 & 5 & \propto & \propto \\ 7 & \mathbf{12} & \propto & 2 \\ \propto & 3 & \propto & \propto \\ 4 & \mathbf{9} & 1 & \propto \end{vmatrix} \quad \begin{vmatrix} 11 & 12 & - & - \\ 21 & \mathbf{212} & - & 24 \\ - & 32 & - & - \\ 41 & \mathbf{412} & 43 & - \end{vmatrix}$$

vertex 2:

$$\begin{vmatrix} 7 & 5 & \propto & \mathbf{7} \\ 7 & 12 & \propto & 2 \\ \mathbf{10} & 3 & \propto & \mathbf{5} \end{vmatrix} \quad \begin{vmatrix} 11 & 12 & - & \mathbf{124} \\ 21 & 212 & - & 24 \\ \mathbf{321} & 32 & - & \mathbf{324} \end{vmatrix}$$

4 9 1 **11** 41 412 43 **4124**

vertex 3:

$$\begin{vmatrix} 7 & 5 & \infty & 7 \\ 7 & 12 & \infty & 2 \\ 10 & 3 & \infty & 5 \\ 4 & \mathbf{4} & 1 & \mathbf{6} \end{vmatrix} \quad \begin{vmatrix} 11 & 12 & - & 124 \\ 21 & 212 & - & 24 \\ 321 & 32 & - & 324 \\ 41 & \mathbf{432} & 43 & \mathbf{4324} \end{vmatrix}$$

vertex 4:

$$\begin{vmatrix} 7 & 5 & \mathbf{8} & 7 \\ \mathbf{6} & \mathbf{6} & \mathbf{3} & 2 \\ \mathbf{9} & 3 & \mathbf{6} & 5 \\ 4 & 4 & 1 & 6 \end{vmatrix} \quad \begin{vmatrix} 11 & 12 & \mathbf{1243} & 124 \\ \mathbf{241} & \mathbf{2432} & \mathbf{243} & 24 \\ \mathbf{3241} & 32 & \mathbf{3243} & 324 \\ 41 & 432 & 43 & 4324 \end{vmatrix}$$

- At 0^{th} iteration it nil give you the direct distances between any 2 nodes

$$D0= \begin{vmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & \infty & 0 & 15 \\ 15 & \infty & 5 & 0 \end{vmatrix}$$

- At 1^{st} iteration we have to check the each pair(i,j) whether there is a path through node 1.if so we have to check whether it is minimum than the previous value and if I is so than the distance through 1 is the value of d1(i,j).at the same time we have to solve the intermediate node in the matrix position p(i,j).

$$D1= \begin{vmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & \mathbf{35} & 0 & 15 \\ 15 & \mathbf{20} & 5 & 0 \end{vmatrix}$$

p[3,2]= 1
p[4,2]= 1

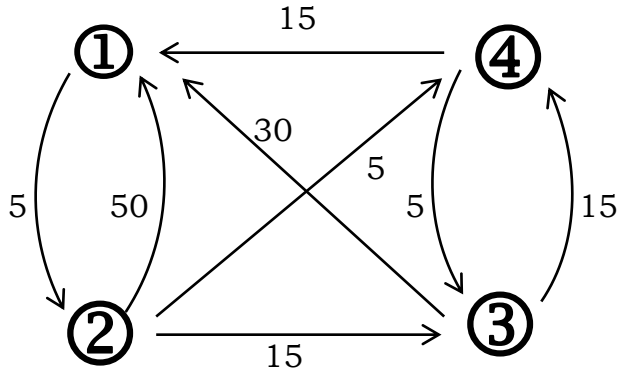


Fig: floyd's algorithm and work

- likewise we have to find the value for N iteration (ie) for N nodes.

$$D2=\begin{vmatrix} 0 & 5 & \mathbf{20} & \mathbf{10} \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{vmatrix} \quad \begin{matrix} P[1,3] = 2 \\ P[1,4] = 2 \end{matrix}$$

$$D3=\begin{vmatrix} 0 & 5 & 20 & 10 \\ \mathbf{45} & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{vmatrix} \quad P[2,1]=3$$

$$D4=\begin{vmatrix} 0 & 5 & \mathbf{15} & 10 \\ 20 & 0 & \mathbf{10} & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{vmatrix} \quad \begin{matrix} P[1,3]=4 \\ P[2,3]=4 \end{matrix}$$

- D4 will give the shortest distance between any pair of nodes.
- If you want the exact path then we have to refer the matrix p.The matrix will be,

$$P=\begin{vmatrix} 0 & 0 & 4 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{vmatrix} \quad 0 \longrightarrow \text{direct path}$$

- Since,p[1,3]=4,the shortest path from 1 to3 passes through 4.
- Looking now at p[1,4]&p[4,3] we discover that between 1 & 4, we have to go to node 2 but that from 4 to 3 we proceed directly.
- Finally we see the trips from 1 to 2, & from 2 to 4, are also direct.
- The shortest path from 1 to 3 is 1,2,4,3.

ALGORITHM :

Function Floyd (L[1..r,1..r]):array[1..n,1..n]
array D[1..n,1..n]

D = L
For k = 1 to n do
For i = 1 to n do
For j = 1 to n do
D [i , j] = min (D[i, j], D[i, k] + D[k, j]
Return D

ANALYSIS:

This algorithm takes a time of $\theta (n^3)$

MULTISTAGE GRAPH

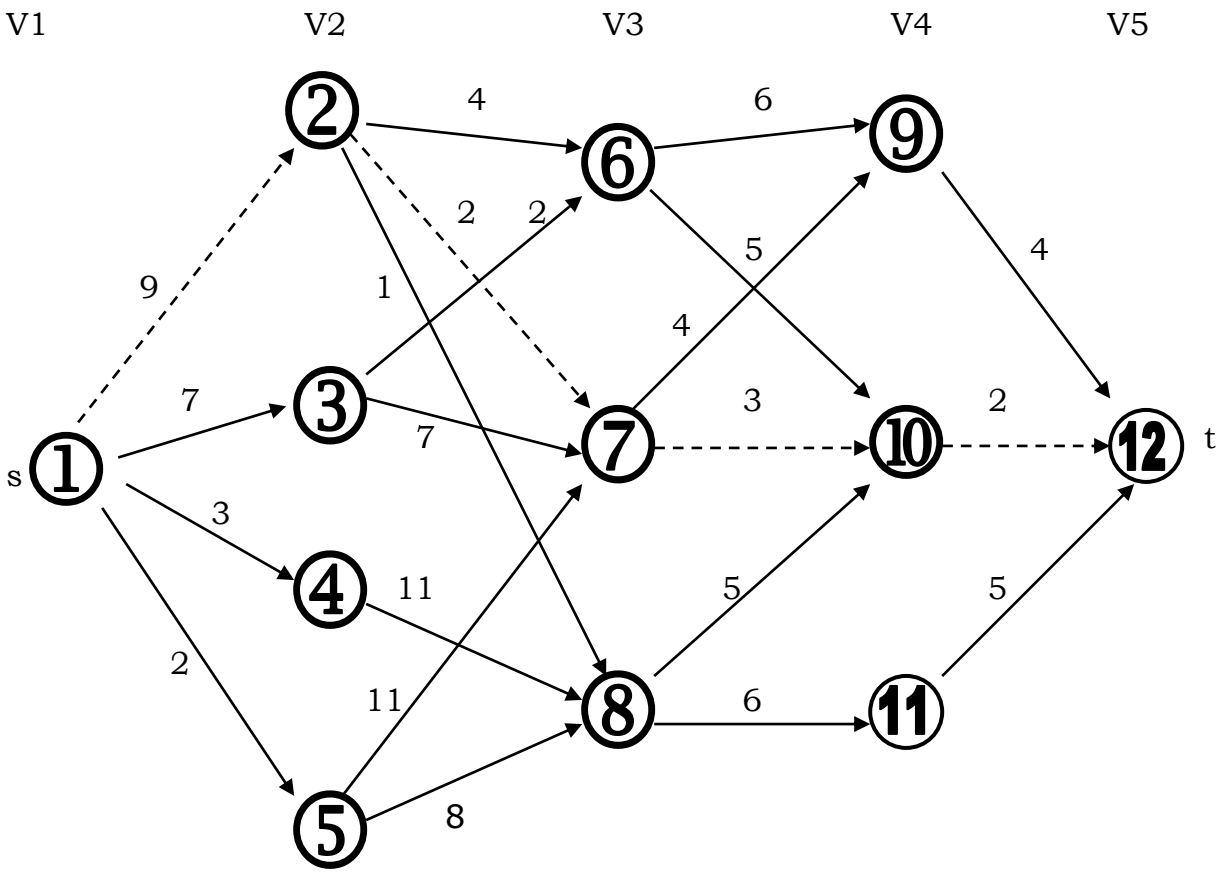
1. A multistage graph G = (V,E) is a directed graph in which the vertices are portioned into K > = 2 disjoint sets Vi, 1 <= i<= k.
2. In addition, if < u,v > is an edge in E, then u < = Vi and V $\sum$ Vi+1 for some i, 1<= i < k.
3. If there will be only one vertex, then the sets Vi and Vk are such that [Vi]=[Vk] = 1.
4. Let 's' and 't' be the source and destination respectively.
5. The cost of a path from source (s) to destination (t) is the sum of the costs of the edger on the path.
6. The *MULTISTAGE GRAPH* problem is to find a minimum cost path from 's' to 't'.
7. Each set Vi 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.
8. This *MULISTAGE GRAPH* problem can be solved in 2 ways.

    a) Forward Method.
    b) Backward Method.

FORWARD METHOD

1. Assume that there are 'k' stages in a graph.
2. In this *FORWARD* approach, we will find out the cost of each and every node starling from the 'k' th stage to the 1^{st} stage.
3. We will find out the path (i.e.) minimum cost path from source to the destination (ie) [Stage-1 to Stage-k].

PROCEDURE:



- Maintain a cost matrix cost (n) which stores the distance from any vertex to the destination.
- If a vertex is having more than one path, then we have to choose the minimum distance path and the intermediate vertex, which gives the minimum distance path, will be stored in the distance array 'D'.
- In this way we will find out the minimum cost path from each and every vertex.
- Finally cost(1) will give the shortest distance from source to destination.
- For finding the path, start from vertex-1 then the distance array D(1) will give the minimum cost neighbour vertex which in turn give the next nearest vertex and proceed in this way till we reach the Destination.
- For a 'k' stage graph, there will be 'k' vertex in the path.
- In the above graph V1…V5 represent the stages. This 5 stage graph can be solved by using forward approach as follows,

STEPS: -		**DESTINATION, D**
Cost (12)=0	⟶	D (12)=0
Cost (11)=5	⟶	D (11)=12
Cost (10)=2	⟶	D (10)=12
Cost (9)=4	⟶	D (9)=12

- For forward approach,

$$\text{Cost}(i,j) = \min_{\substack{l \in Vi+1 \\ (j,l) \in E}} \{C(j,l) + \text{Cost}(i+1,l)\}$$

Cost(8) = min {C (8,10) + Cost (10), C (8,11) + Cost (11) }
= min (5 + 2, 6 + 5)
= min (7,11)
= 7
cost(8) =7 =>D(8)=10

cost(7) = min(c (7,9)+ cost(9),c (7,10)+ cost(10))
(4+4,3+2)
= min(8,5)
= 5
cost(7) = 5 =>D(7) = 10

cost(6) = min (c (6,9) + cost(9),c (6,10) +cost(10))
= min(6+4 , 5 +2)
= min(10,7)
= 7
cost(6) = 7 =>D(6) = 10

cost(5) = min (c (5,7) + cost(7),c (5,8) +cost(8))
= min(11+5 , 8 +7)
= min(16,15)
= 15
cost(5) = 15 =>D(5) = 18

cost(4) = min (c (4,8) + cost(8))
= min(11+7)
= 18
cost(4) = 18 =>D(4) = 8

cost(3) = min (c (3,6) + cost(6),c (3,7) +cost(7))
= min(2+7 , 7 +5)
= min(9,12)
= 9
cost(3) = 9 =>D(3) = 6

cost(2) = min (c (2,6) + cost(6),c (2,7) +cost(7) ,c (2,8) +cost(8))
= min(4+7 , 2+5 , 1+7)
= min(11,7,8)
= 7

cost(2) = 7 =>D(2) = 7

cost(1) = min (c (1,2)+cost(2) ,c (1,3)+cost(3) ,c (1,4)+cost(4) ,c(1,5)+cost(5))
= min(9+7 , 7 +9 , 3+18 , 2+15)
= min(16,16,21,17)
= 16
cost(1) = 16 =>D(1) = 2

⟶ The path through which you have to find the shortest distance.

(i.e.) ① ⟶ ② ⟶ ⑦ ⟶ ⑩ ⟶ ⑫

Start from vertex - 2

D (1) = 2
D (2) = 7
D (7) = 10
D (10) = 12

So, the minimum –cost path is,

① —9→ ② —2→ ⑦ —3→ ⑩ —2→ ⑫

∴ The cost is 9+2+3+2+=16

ALGORITHM: FORWARD METHOD

Algorithm FGraph (G,k,n,p)

```
//  The I/p is a k-stage graph G=(V,E) with 'n' vertex.
//  Indexed in order of stages E is a set of edges.
// and c[i,J] is the cost of<i,j>,p[1:k] is a minimum cost path.
{
     cost[n]=0.0;
      for j=n-1 to 1 step-1 do
  {
     //compute cost[j],
    //   let 'r' be the vertex such that <j,r> is an edge of 'G' &
   //   c[j,r]+cost[r] is minimum.
```

```
    cost[j] = c[j+r] + cost[r];
     d[j] =r;
   }
  // find a minimum cost path.

       P[1]=1;
       P[k]=n;
       For j=2 to k-1 do
       P[j]=d[p[j-1]];
}
```

ANALYSIS:

The time complexity of this forward method is O($|V| + |E|$)

BACKWARD METHOD

- if there one 'K' stages in a graph using back ward approach. we will find out the cost of each & every vertex starting from 1st stage to the kth stage.
- We will find out the minimum cost path from destination to source (ie)[from stage k to stage 1]

PROCEDURE:

1. It is similar to forward approach, but differs only in two or three ways.
2. Maintain a cost matrix to store the cost of every vertices and a distance matrix to store the minimum distance vertex.
3. Find out the cost of each and every vertex starting from vertex 1 up to vertex k.
4. To find out the path star from vertex 'k', then the distance array D (k) will give the minimum cost neighbor vertex which in turn gives the next nearest neighbor vertex and proceed till we reach the destination.

STEP:

Cost(1) = 0 => D(1)=0
Cost(2) = 9 => D(2)=1
Cost(3) = 7 => D(3)=1
Cost(4) = 3 => D(4)=1
Cost(5) = 2 => D(5)=1

Cost(6) =min(c (2,6) + cost(2),c (3,6) + cost(3))
=min(13,9)

cost(6) = 9 =>D(6)=3

Cost(7) =min(c (3,7) + cost(3),c (5,7) + cost(5) ,c (2,7) + cost(2))

=min(14,13,11)

cost(7) = 11 =>D(7)=2

Cost(8) =min(c (2,8) + cost(2),c (4,8) + cost(4) ,c (5,8) +cost(5))
=min(10,14,10)

cost(8) = 10 =>D(8)=2

Cost(9) =min(c (6,9) + cost(6),c (7,9) + cost(7))
=min(15,15)

cost(9) = 15 =>D(9)=6

Cost(10)=min(c(6,10)+cost(6),c(7,10)+cost(7)),c (8,10)+cost(8)) =min(14,14,15)
cost(10)= 14 =>D(10)=6

Cost(11) =min(c (8,11) + cost(8))

cost(11) = 16 =>D(11)=8

cost(12)=min(c(9,12)+cost(9),c(10,12)+cost(10),c(11,12)+cost(11))
=min(19,16,21)

cost(12) = 16 =>D(12)=10

PATH:

Start from vertex-12
D(12) = 10
D(10) = 6
D(6) = 3
D(3) = 1

So the minimum cost path is,

$1 \xrightarrow{7} 3 \xrightarrow{2} 6 \xrightarrow{5} 10 \xrightarrow{2} 12$

The cost is 16.

ALGORITHM : BACKWARD METHOD

Algorithm BGraph (G,k,n,p)

// The I/p is a k-stage graph G=(V,E) with 'n' vertex.
// Indexed in order of stages E is a set of edges.

```
// and c[i,J] is the cost of<i,j>,p[1:k] is a minimum cost path.
{
     bcost[1]=0.0;
      for j=2 to n do
   {
      //compute bcost[j],
     //   let 'r' be the vertex such that <r,j> is an edge of 'G' &
    //   bcost[r]+c[r,j] is minimum.

    bcost[j] = bcost[r] + c[r,j];
     d[j] =r;
   }
  // find a minimum cost path.

        P[1]=1;
        P[k]=n;
        For j= k-1 to 2 do
        P[j]=d[p[j+1]];
}
```

TRAVELLING SALESMAN PROBLEM

- Let G(V,E) be a directed graph with edge cost c_{ij} is defined such that $c_{ij} > 0$ for all i and j and $c_{ij} = \propto$,if <i,j> $\notin$ E.
  Let V ǂn and assume n>1.
- The traveling salesman problem is to find a tour of minimum cost.
- A tour of G is a directed cycle that include every vertex in V.
- 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 (ie) 1.

APPLICATION :

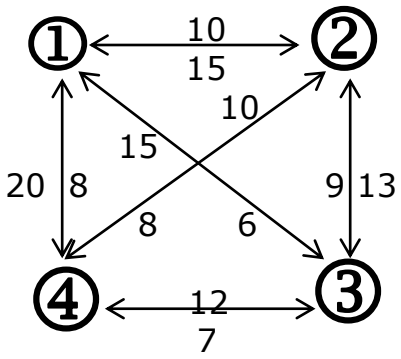
1. Suppose we have to route a postal van to pick up mail from the mail boxes located at 'n' different sites.
2. An n+1 vertex graph can be used to represent the situation.
3. One vertex represent the post office from which the postal van starts and return.
4. Edge <i,j> is assigned a cost equal to the distance from site 'i' to site 'j'.
5. the route taken by the postal van is a tour and we are finding a tour of minimum length.
6. every tour consists of an edge <1,k> for some k $\in$ V-{} and a path from vertex k to vertex 1.
7. the path from vertex k to vertex 1 goes through each vertex in V-{1,k} exactly once.
8. the function which is used to find the path is

   $g(1,V-\{1\}) = \min\{ c_{ij} + g(j,s-\{j\})\}$

9. 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.
10. the function g(1,v-{1}) is the length of an optimal tour.

STEPS TO FIND THE PATH:

1. Find $g(i,\Phi) = c_{i1}$, $1<=i<n$, hence we can use equation(2) to obtain g(i,s) for all s to size 1.
2. That we have to start with s=1,(ie) there will be only one vertex in set 's'.
3. Then s=2, and we have to proceed until |s| <n-1.
4. for example consider the graph.



Cost matrix

$$\begin{vmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{vmatrix}$$

g(i,s) → set of nodes/vertex have to visited.

↓

starting position

$g(i,s) = \min\{c_{ij} + g(j,s-\{j\})$

STEP 1:

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

min{10+25,15+25,20+23}
min{35,35,43}
=35

STEP 2:

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

$\min\{9+20,10+15\}$
$\min\{29,25\}$
$=25$

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

$\min\{13+18,12+13\}$
$\min\{31,25\}$
$=25$

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

$\min\{8+15,9+18\}$
$\min\{23,27\}$
$=23$

STEP 3:

1. $g(3,\{4\}) = \min\{c_{34} + g\{4,\Phi\}\}$
   $12+8 = 20$

2. $g(4,\{3\}) = \min\{c_{43} + g\{3,\Phi\}\}$
   $9+6 = 15$

3. $g(2,\{4\}) = \min\{c_{24} + g\{4,\Phi\}\}$
   $10+8 = 18$

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

5. $g(2,\{3\}) = \min\{c_{23} + g\{3,\Phi\}\}$
   $9+6=15$

6. $g(3,\{2\}) = \min\{c_{32} + g\{2,\Phi\}\}$
   $13+5=18$

STEP 4:

$g\{4,\Phi\} = c_{41} = 8$

$g\{3,\Phi\} = c_{31} = 6$

$g\{2,\Phi\} = c_{21} = 5$

$|s| = 0.$

i =1 to n.

$g(1,\Phi) = c_{11} \Rightarrow 0$

$g(2,\Phi) = c_{21} \Rightarrow 5$

$g(3,\Phi) = c_{31} \Rightarrow 6$

$g(4,\Phi) = c_{41} \Rightarrow 8$

$|s| = 1$

i =2 to 4

$g(2,\{3\}) = c_{23} + g(3,\Phi)$
$= 9+6 = 15$

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

$g(3,\{2\}) = c_{32} + g(2,\Phi)$
$= 13+5 = 18$

$g(3,\{4\}) = c_{34} + g(4,\Phi)$
$= 12+8 = 20$

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

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

$|s| = 2$

$i \neq 1$, $1 \in s$ and $i \in s$.

$g(2,\{3,4\}) = \min\{c_{23}+g(3\{4\}), c_{24}+g(4,\{3\})\}$
$\min\{9+20, 10+15\}$
$\min\{29,25\}$

=25

$g(3,\{2,4\}) = \min\{c_{32}+g(2\{4\}), c_{34}+g(4,\{2\})\}$
$\min\{13+18, 12+13\}$
$\min\{31,25\}$
$=25$

$g(4,\{2,3\}) = \min\{c_{42}+g(2\{3\}), c_{43}+g(3,\{2\})\}$
$\min\{8+15, 9+18\}$
$\min\{23,27\}$
$=23$

$|s| = 3$

$g(1,\{2,3,4\}) = \min\{c_{12}+g(2\{3,4\}), c_{13}+g(3,\{2,4\}), c_{14}+g(4,\{2,3\})\}$
$\min\{10+25, 15+25, 20+23\}$
$\min\{35,35,43\}$
$=35$

optimal cost is 35

the shortest path is,

$g(1,\{2,3,4\}) = c_{12} + g(2,\{3,4\})$ => 1->2

$g(2,\{3,4\}) = c_{24} + g(4,\{3\})$ => 1->2->4

$g(4,\{3\}) = c_{43} + g(3\{\Phi\})$ => 1->2->4->3->1

so the optimal tour is **1 → 2 → 4 →3 → 1**