UNIT-III – DYNAMIC PROGRAMMING

Dynamic Programming: General method, Applications- Matrix chain multiplication, Optimal binary search trees, 0/1 knapsack problem, All pairs shortest path problem, Travelling sales person problem.

-0-0-0-0-

INTRODUCTION

The drawback of greedy method is, we will make one decision at a time. This can be overcome in dynamic programming. In this we will make more than one decision at a time.

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. Dynamic programming is applicable when the sub-problems are not independent, that is when sub-problems share sub-sub-problems. A dynamic programming algorithm solves every sub-sub-problem just once and then saves its answer in a table, there by avoiding the work of re-computing the answer every time the sub-problem is encountered.

Definition:

It is a programming technique in which solution is obtained from a sequence of decisions

General Method:

The fundamental dynamic programming model may be written as,

$$F_n(R) = \max_{0 \le R_n \le R} \{ P_n(R) + F_{n-1}(R - R_n) \}$$

Where n = 2,3,4…

$F_n(0)=0$

$F_1(R)=P_1(R)$

Once $F_1(R)$ is known equation(1) provides a relation for evaluation of $F_2(R)$, $F_3(R)$…. This recursive process ultimately leads to the value of $F_{n-1}(R)$ and finally $F_n(R)$ at which process stops.

A dynamic programming problem can be divided into a number of stages where an optimal decision must be made at each stage. The decision made at each stage must take into account its effects not only on the next stage, but also on the entire subsequent stages. Dynamic programming provides a systematic procedure whereby starting with the last stage of the problem and working backwards one makes an optimal decision for each stage of problem. The information for the last stage is the information derived from the previous stage.

Dynamic programming design involves 4 major steps.

1) Characterize the structure of optimal solution.
2) Recursively define the value of an optimal solution.
3) Compute the value of an optimum solution in a bottom up fashion.
4) Construct an optimum solution from computed information.

The Dynamic programming technique was developed by Bellman based upon his principle known as principle of optimality. This principle states that “An optimal policy has the property that, what ever the initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision”.

General Characteristics of Dynamic Programming:

The general characteristics of Dynamic programming are

1) The problem can be divided into stages with a policy decision required at each stage.
2) Each stage has number of states associated with it.
3) Given the current stage an optimal policy for the remaining stages is independent of the policy adopted.
4) The solution procedure begins be finding the optimal policy for each state of the last stage.
5) A recursive relation is available which identifies the optimal policy for each stage with n stages remaining given the optimal policy for each stage with (n-1) stages remaining.

APPLICATIONS OF DYNAMIC PROGRAMMING

1) **Matrix Chain Multiplication**
2) **Optimal Binary search Trees**
3) **0/1 Knapsack Problem**
4) **Multi stage Graph**
5) **Traveling sales person problem**
6) **Reliability Design**

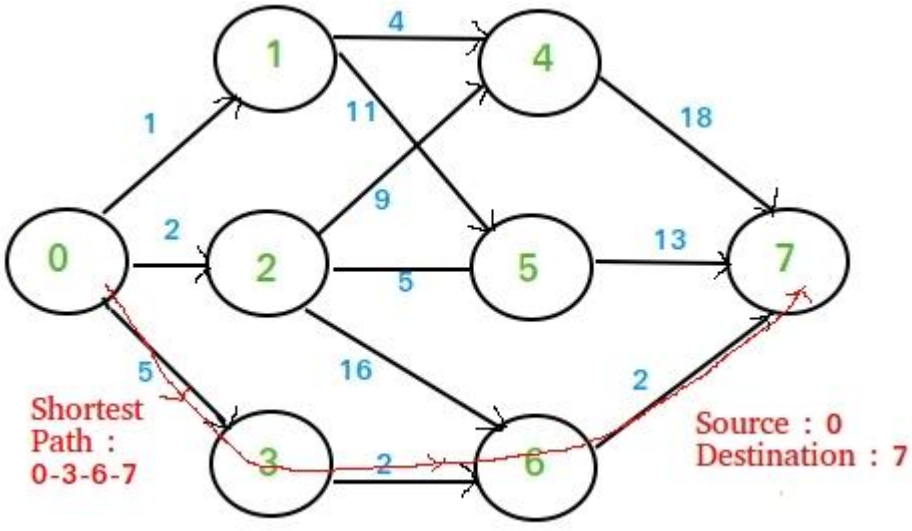
Multistage Graph (Shortest Path)

A **Multistage graph** is a directed, weighted graph in which the nodes can be divided into a set of stages such that all edges are from a stage to next stage only (In other words there is no edge between vertices of same stage and from a vertex of current stage to previous stage).

The vertices of a multistage graph are divided into n number of disjoint subsets S = { S1 , S2 , S3 **……..** Sn }, where S1 is the source and Sn is the sink (destination). The cardinality of S1 and Sn are equal to 1. i.e., |S1| = |Sn| = 1.

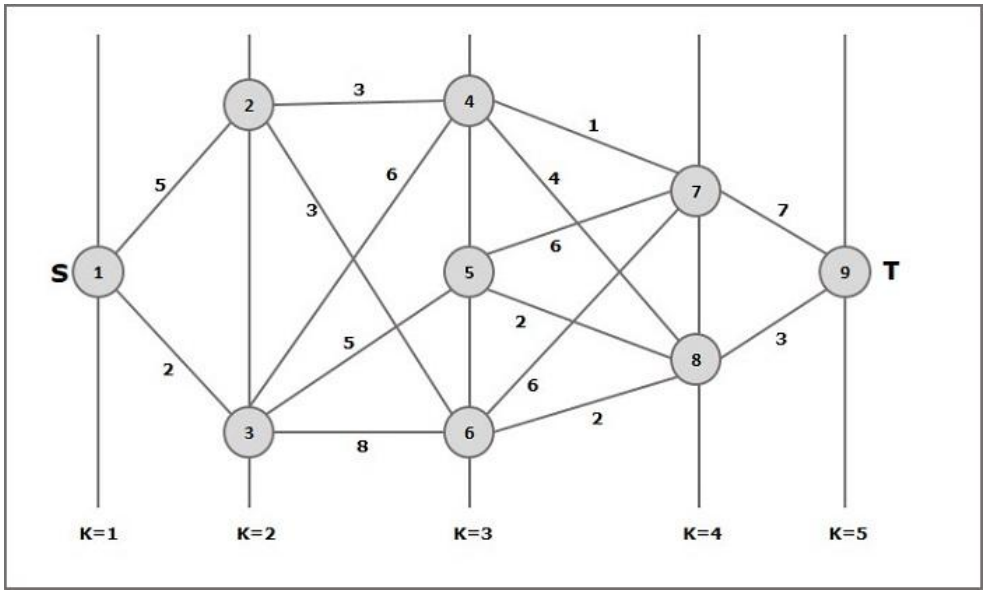
We are given a multistage graph, a source and a destination, we need to find shortest path from source to destination. By convention, we consider source at stage 1 and destination as last stage.

Following is an example graph we will consider in this article :-



Example

Consider the following example to understand the concept of multistage graph.



According to the formula, we have to calculate the cost **(i, j)** using the following steps

Step 1: Cost (K-2, j)

In this step, three nodes (node 4, 5. 6) are selected as **j**. Hence, we have three options to choose the minimum cost at this step.

Cost(3, 4) = min {c(4, 7) + Cost(7, 9),c(4, 8) + Cost(8, 9)} = 7

Cost(3, 5) = min {c(5, 7) + Cost(7, 9),c(5, 8) + Cost(8, 9)} = 5

Cost(3, 6) = min {c(6, 7) + Cost(7, 9),c(6, 8) + Cost(8, 9)} = 5

Step 2: Cost (K-3, j)

Two nodes are selected as j because at stage k - 3 = 2 there are two nodes, 2 and 3. So, the value i = 2 and j = 2 and 3.

Cost(2, 2) = min {c(2, 4) + Cost(4, 8) + Cost(8, 9),c(2, 6) +

Cost(6, 8) + Cost(8, 9)} = 8

Cost(2, 3) = {c(3, 4) + Cost(4, 8) + Cost(8, 9), c(3, 5) + Cost(5, 8)+ Cost(8, 9), c(3, 6) + Cost(6, 8) + Cost(8, 9)} = 10

Step 3: Cost (K-4, j)

Cost (1, 1) = {c(1, 2) + Cost(2, 6) + Cost(6, 8) + Cost(8, 9), c(1, 3) + Cost(3, 5) + Cost(5, 8) + Cost(8, 9))} = 12

c(1, 3) + Cost(3, 6) + Cost(6, 8 + Cost(8, 9))} = 13

Hence, the path having the minimum cost is **1**→ **3**→ **5**→ **8**→ **9**.

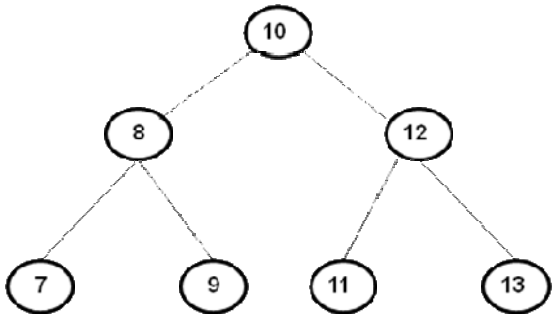
1) OPTIMAL BINARY SEARCH TREE

A Binary search tree is a binary tree that is either empty or in which every node contains a key and satisfies the following conditions.

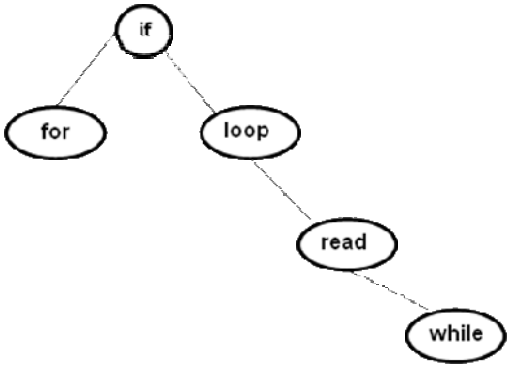
1. The key in the left child of a node (if it exists) is less than the key in its parent node.
2. The key in the right child of a node is greater than the key in its parent node.
3. The left and right subtrees of the root are again binary search trees.

No two entries in a binary search tree may have equal keys.

Ex) Binary Search Trees



(1)



(2)

If we want to search an element in binary search tree, first that element is compared with root node. If element is less than root node then search continue in left subtree. If element is greater than root node then search continue in right subtree. If element is equal to root node then print search was successful (element found) and terminate search procedure.

```
algorithm search(t,x)
{
    if (t==0) then return 0;
    else
        if  (x = t->data) then return t;
        else
            if ( x< t->data) then return search (t->left, x);
            else
                  return search (t->right, x);
}
```

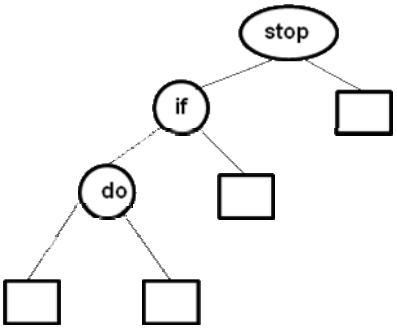
The possible binary search trees for the identifier set (a1,a2,a3=do, if, stop)
Hence n=3

The number of possible binary search trees = $\frac{1}{n+1}{}^{2n}C_n = \frac{1}{3+1}{}^{2x3}C_3$

$= \frac{1}{4}{}^{6}C_3 = \frac{1}{4}\times\frac{1\times2\times3\times4\times5\times6}{1\times2\times3\times1\times2\times3} = \frac{1}{4}\times 20 = 5$

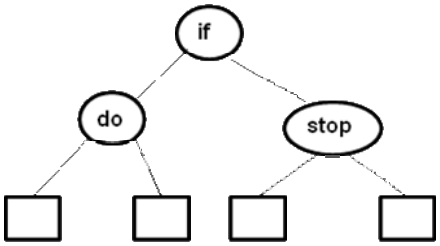
The sequence of key words to construct a BST may be one of the following

stop,if,do



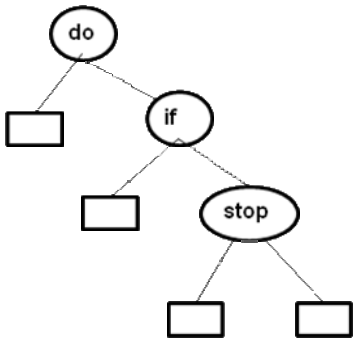
(a)

if,do,stop



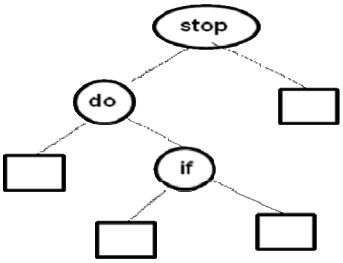
(b)

do,if,stop



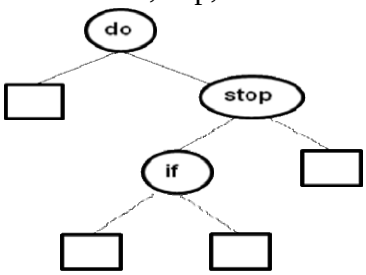
(c)

stop,do,if



(d)

do,stop,if



(e)

$$\cos t(T) = \sum_{1 \le i \le n} p(i) \times level(a_i) + \sum_{0 \le i \le n} q(i) \times level(E_i - 1)$$

$$p(i) = q(i) = \frac{1}{7}$$

$$a)\ 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 3 \times \frac{1}{7} = \frac{15}{7}$$

$$b)\ 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 2 \times \frac{1}{7} + 2 \times \frac{1}{7} + 2 \times \frac{1}{7} + 2 \times \frac{1}{7} + 2 \times \frac{1}{7} = \frac{13}{7}$$

$$c)\ 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 3 \times \frac{1}{7} = \frac{15}{7}$$

$$d)\ 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 3 \times \frac{1}{7} = \frac{15}{7}$$

$$e)\ 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 1 \times \frac{1}{7} + 2 \times \frac{1}{7} + 3 \times \frac{1}{7} + 3 \times \frac{1}{7} = \frac{15}{7}$$

In the above binary search tree the cost of the tree 2 is minimum. Hence it is optimal binary search tree.

A binary search tree have maximum of two childs. i.e 0, 1 or 2 childs.
p(i) - probability of searching an internal node.
Q(i) – probability of searching an external node.
A successful search is terminated at an internal node denoted by circle (O) ans unsuccessful search is terminated at an external node by square (□).

$$\cos t(T) = \sum_{1 \le I \le n} p(i) \times level(a_i) + \sum_{0 \le I \le n} q(i) \times level(E_i - 1)$$

Example 2) Let n=4 and (a1, a2, a3, a4) = (do, if , read, while) -- internal nodes
p(1:4)=(3,3,1,1)
q(0:4)=(2,3,1,1,1)

External nodes (E_0,E_1,E_2,E_3,E_4)
As there are 4 internal nodes (a1, a2, a3, a4) = (do, if , read, while)
the number of possible binary search trees = $\frac{1}{n+1} {}^{2n}C_n = 14$

w(i,j) = p(j) + q(j) + w(i,j-1) weight of t_{ij}

$$c(i, j) = \min_{i<k\le j} \{ c(i, k-1) + c(k, j) \} + w(i, j)$$ cost of t_{ij}

r(i,j) = k root of t_{ij}

W_{01} = E_0 a_1 E_1
W_{12} = E_1 a_2 E_2
W_{23} = E_2 a_3 E_3
W_{34} = E_3 a_4 E_4
W_{02} = E_0 a_1 E_1 a_2 E_2
W_{13} = E_1 a_2 E_2 a_3 E_3
W_{24} = E_2 a_3 E_3 a_4 E_4
W_{03} = E_0 a_1 E_1 a_2 E_2 a_3 E_3
W_{14} = E_1 a_2 E_2 a_3 E_3 a_4 E_4
W_{04} = $E_0a_1E_1a_2E_2a_3E_3a_4E_4$

In order to solve above problem, first we will draw one table by taking 'I' corresponds to columns. The cells in the table can be indicated as $w_{j,j+i}$, $c_{j,j+i}$,$R_{j,j+i.}$

	0	1	2	3	4
0	W_{00}=2 C_{00}=0 R_{00}=0	W_{11}=3 C_{11} =0 R_{11}=0	W_{22}=1 C_{22}=0 R_{22}=0	W_{33}=1 C_{33}=0 R_{33}=0	W_{44}=1 C_{44}=0 R_{44}=0
1	W_{01}= C_{01}= R_{01}=	W_{12}= C_{12} = R_{12} =	W_{23}= C_{23}= R_{23}=	W_{34}= C_{34} = R_{34} =	
2	W_{02}= C_{02} = R_{02} =	W_{13}= C_{13} = R_{13} =	W_{24}= C_{24}= R_{24}=		
3	W_{03}= C_{03} = R_{03} =	W_{14}= C_{14} = R_{14} =			
4	W_{04}= C_{04} = R_{04} =				

Principle of optimality was applied.

The first row is initiated as W(i,i)=q(i) R(i,i)=0 C(i,i)) = 0

It means

$W_{00} = q(0) = 2$ $C_{00} = R_{00} = 0$

$W_{11} = q(1) = 3$ $C_{11} = R_{11} = 0$

$W_{22} = q(2) = 1$ $C_{22} = R_{22} = 0$

$W_{33} = q(3) = 1$ $C_{33} = R_{33} = 0$

$W_{44} = q(4) = 1$ $C_{44} = R_{44} = 0$

The remaining values of cells can be calculated using equations as follows

w(i,j)=p(j) + q(j) + w(i,j-1)

w(0,1)=p(1) + q(1) + w(0,0) = 3+3+2 = 8

w(1,2) = p(2) + q(2) + w(1,1) = 3+1+3 = 7

w(2,3) = p(3) + q(3) + w(2,2) = 1+1+1 = 3

w(3,4) = p(4) + q(4) + w(3,3) = 1+1+1 = 3

w(0,2) = p(2) + q(2) + w(0,1) = 3+1+8 = 12

w(1,3) = p(3) + q(3) + w(1,2) = 1+1+7 = 9

w(2,4) = p(4) + q(4) + w(2,3) = 1+1+3 = 5

w(0,3) = p(3) + q(3) + w(0,2) = 1+1+12 = 14

w(1,4) = p(4) + q(4) + w(1,3) = 1+1+9 = 11

w(0,4) = p(4) + q(4) + w(0,3) = 1+1+14 = 16

$$c(i, j) = \min_{i<k\le j} \{ c(i, k-1) + c(k, j) \} + w(i, j)$$

C(0,1) = min { c(0,0) + c(1,1) } + w(0,1) = 0 + 0 + 8 = 8 when k=1

R(0,1)=1

C(1,2) = min { c(1,1) + c(2,2) } + w(1,2) = 0 + 0 + 7 = 7 when k=2

R(1,2)=2

C(2,3) = min { c(2,2) + c(3,3) } + w(2,3) = 0 + 0 + 3 = 3 when k=3

R(2,3)=3

C(3,4) = min { c(3,3) + c(4,4) } + w(3,4) = 0 + 0 + 3 = 3 when k=4

R(3,4)=4

When k=2

C(0,2) = min { c(0,1) + c(2,2) } + w(0,2) = 8 + 0 + 12 = 20

When k=1

C(0,2) = min { c(0,0) + c(1,2) } + w(0,2) = 0 + 7 + 12 = 19

C(0,2) = min{20,19} = 19

r(0, 2) = 1

When k=3 or 2

C(1,3) = min { c(1,1) + c(2,3), c(1,2) + c(3,3) } + w(1,3) ={ 0 + 3, 7 + 0} + 9 = 12

r(1, 3) = 2

When k=3

C(2,4) = min { c(2,2) + c(3,4) } + w(2,4) = 0 + 3 + 5 = 8

When k=4

C(2,4) = min { c(2,3) + c(4,4) } + w(2,4) = 3 + 0 + 5 = 8

r(2,4)=3

When k=1 or 2 or 3
C(0,3) = min { c(0,0) + c(1,3), c(0,1) + c(2,3), c(0,2)+c(3,3) } + w(1,3)
=min{12,11,10} + 14 = 10 + 14 = 24
r(0,3)=2

When k=2 or 3 or 4
C(1,4) = min { c(1,1) + c(2,4), c(1,2) + c(3,4), c(1,3)+c(4,4) } + w(1,4)
=min{0+8, 7+3, 12+0} + 11 = 8 + 11 = 19
r(1,4)=2

When k=1 or 2 or 3 or 4
C(1,4) = min { c(0,0) + c(1,4), c(0,1) + c(2,4), c(0,2)+c(3,4), c(0,3)+c(4,4) } + w(0,4)
=min{19, 16, 23, 25} + 16 = 16 + 16 = 32
R(0,4)=2

	0	1	2	3	4
0	W_{00}=2 C_{00}=0 R_{00}=0	W_{11}=3 C_{11}=0 R_{11}=0	W_{22}=1 C_{22}=0 R_{22}=0	W_{33}=1 C_{33}=0 R_{33}=0	W_{44}=1 C_{44}=0 R_{44}=0
1	W_{01}=8 C_{01}=8 R_{01}=1	W_{12}=7 C_{12}=7 R_{12}=2	W_{23}=3 C_{23}=3 R_{23}=3	W_{34}=3 C_{34}=3 R_{34}=4	
2	W_{02}=12 C_{02}=19 R_{02}=1	W_{13}=9 C_{13}=12 R_{13}=2	W_{24}=5 C_{24}=8 R_{24}=3		
3	W_{03}=14 C_{03}=25 R_{03}=2	W_{14}=11 C_{14}=19 R_{14}=2			
4	W_{04}=16 C_{04}=32 R_{04}=2				

Now observe the tables last cell i.e. 4^{th} row 0^{th} column, contains $r_{04} = 2$ i.e. $r_{04} = a_2$ (2 corresponds to second node a_2).
R04=k, then k=2
To build OBST R(0,4)=2=k=2

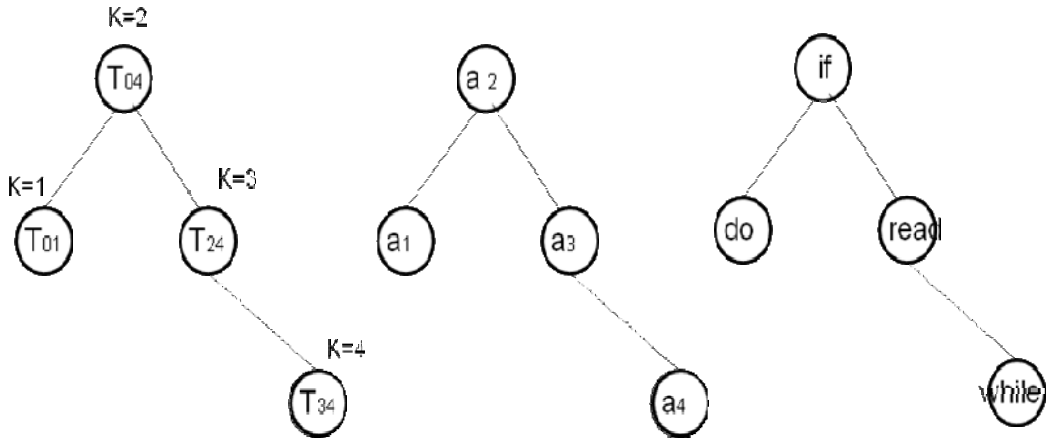
Hence a_2 become the root node
Let T be OBST $T_{i,j} = T_{i,k-1}$, and $T_{k,j}$

T_{04} is divided into two parts i.e T01 and T24
T_{01}=r(0,1)=1=k=1
T_{24}=r(2,4)=3=k=3

T_{01} is divided into two parts T_{00} and T_{11} where k=1
T_{24} is divided into two parts T_{22} and T_{34} where k=3
T_{34} is divided into two parts T_{33} and T_{44} where k=4

Since r00, r11, r22, r33, r44 = 0 these are external nodes and can be neglected.

Let T be the optimal binary search tree




```
Algorithm OBST(p,q,n)
{
  for i:=0 to n-1 do
  {
     w[i,i]:=q[i];
      r[i,i]:=0;
      c[i,i]:=0;
      w[i,i+1]:=q[i]+q[i+1]+p[i+1];
      r[i,i+1]:=i+1;
     c[i,i+1]=q[i]+q[i+1]+p[i+1];
   }
      w[n,n]:=q[n];
       r[n,n]:=0;
       c[n,n]:=0.0;
for m:=2 to n do
   for i:=0 to n-m do
      {
          j:=i+m;
          w[i, j]:=w[i, j-1]+p[j]+q[j];
          k:=find(c, r, i, j);
          c[i, j]:=w[i, j]+c[i, k-1]+c[k, j];
          r[i,j]:=k;
      }
 write(c[0,n],w[0,n],r[0,n];
}

algorithm find(c, r, i, j)
{
    min := ∞;
    for m := r[i, j-1] to r[i+1, j] do
        if ((c[i, m-1]+c[m, j]) < min) then
         {
              min:=c[i, m-1]+c[m, j];
             l:=m;
         }
  return l;
}
```

Time complexity O(n log(n))

2) 0/1 KNAPSACK PROBLEM

In this problem, 'n' objects are given. with each object 'i' having a weight of 'w_i' & a knapsack having a capacity of 'm' is given. If an object 'i' is placed in the knapsack, a profit of 'P_ix_i' is earned A solution to this knapsack problem can be obtained by making a sequence of decisions on the variables x_1, x_2, x_3,… x_n. A decision of variable x_i involves in determining which of the values 0 or 1 is to be assigned to it.

Following a decision on any object x_i. We may be in any of 2 possible states.

1) The capacity remaining in the knapsack is m and no profit has earned. (object=0)
2) The capacity remaining is m-w_i and a profit of p_i has earned.

It is clear that the remaining decisions x_{i+1}, x_{i+2},….x_n must be optimal w.r.t. the problem state resulting from the decision of x_i (i=1). Hence the principle of optimality holds.

Suppose that a store contains different types of ornaments, which are made up of gold.

Let n1, n2, n3 be ornaments, cost and weight of these ornaments are c1, c2, c3 dollars w1, w2, w3 pounds respectively. Now a thief wants to rob the ornaments such that he should get maximum profit. In this the thief can't place fraction of ornament in the bag, i.e. either he can place ornament completely in the bag or he can't place ornament. So

Xi= 0 or 1.

Xi = 0 means we can not place ornament in the bag.

Xi = 1 means we can place ornament completely in the bag.

This problem contains either 0 or 1, hence the problem is called 0/1 Knapsack problem.

Example : Consider the knapsack instance n=3, (w1, w2, w3) = (2, 3, 4), (p1, p2, p3) = (1, 2, 5) and m=6

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

$s^0 = \{0,0\}$

$s_1^i = s^{i-1} + (p_i, w_i)$

$s_1^1 = s^0 + (p_1, w_1)$ ----------addition

$=\{(0,0)\}+\{(1,2)\}$

$s_1^1 = \{(1,2)\}$

$s^1 = s^0 + s_1^1$ ----------merging s^0 *and* s_1^0 *results* s^1

$s^1 = \{(0,0)\}+\{(1,2)\} = \{(0,0)(1,2)\}$

$s_1^2 = s^1 + (p_2, w_2)$----------addition

$=\{(0,0),(1,2)\}+\{2,3\}$

$= \{(2,3),(3,5)\}$

$s^2 = s^1 + s_1^2$----------merging

$s^2 = \{(0,0),(1,2)\}+\{(2,3),(3,5)\}$

$s^2 = \{(0,0),(1,2),(2,3),(3,5)\};$

$s_1^3 = s^2 + (p_3, w_3)$ ---------addition

$= \{(0,0),(1,2),(2,3),(3,5)\} + \{5,4\}$

$= \{(5,4),(6,6),(7,7),(8,9)\}$

$s^3 = s^2 + s_1^3$ ---------- merging

$s^3 = \{(0,0),(1,2),(2,3),(3,5)\} + \{(5,4),(6,6),(7,7),(8,9)\}$

$s^3 = \{(0,0),(1,2),(2,3),(3,5),(5,4),(6,6),(7,7),(8,9)\};$

Using purge rule(dominance rule) in the set S^3 on ordered pairs (3,5) (5,4) i.e. 3<5 and 5>4. so we can eliminate (3,5) from S^3 .

As knapsack maximum capacity is 6 we can eliminate (7,7),(8,9) from S^3.
Now S^3 = {(0,0),(1,2),(2,3),(5,4),(6,6)}.

After applying purge rule we will check the following condition inorder to find solution

If $(p_i, w_i) \in s^n$ *and* $(p_i, w_i) \notin s^{n-1}$ *then* $x_n = 1$ *otherwise* $x_n = 0$

$(6,6) \in s^3$ *and* $(6,6) \notin s^2$ *so* $x_3 = 1$ $\quad (6,6) - (5,4) = (1,2)$

$(1,2) \in s^2$ $\quad (1,2) \notin s^1 \rightarrow$ *False* $\quad x_2 = 0$

$(1,2) \in s^1$ $\quad (1,2) \notin s^0 \rightarrow$ *True* $\quad x_1 = 1$

$\therefore x_1 = 1, x_2 = 0, x_3 = 1$

Maximum profit is $\sum p_i x_i = p_1x_1 + p_2x_2 + p_3x_3 = 1\times1 + 2\times0 + 5\times1 = 1+5 = 6$

Optimal solution is (x1,x2,x3) = (1,0,1)

Max Profit is $\sum p_i x_i = 6$

3) MULTISTAGE GRAPH: A multi stage Graph G=(V,E) is a directed graph in which the vertices are partitioned into k>=2 disjoint sets, V_i where 1<=i<=k. If <u,v> is an edge in E, then uϵV_i, vϵV_{i+1} where 1<=i<=k. The sets v_i and vk are such that $|v_i| = |v_k| = 1$. Let s and T be 2 vertices in v_i and v_k respectively then the vertex s is called the source and T is called the Sink" or "Destination". Let c(i,j) be the cost of the edge <i,j>. The cost of a path from S to T is the sum of the costs of edges on the path. The multistage graph problem is to find minimum cost path from S to T. Each set Vi defines a stage in the graph. Because of the constraints on E, every path from s to t starts in stage 1, goes to stage2, then stage 3, then to stage 4 and so on. And eventually terminates in stage k. figure shows a five stage graph. A minimum cost s to t path is indicated by broken edges.

Consider a resource allocation problem in which n units of resources are to be allocated to r projects. If j 0<=j<=n, units of the resource are allocated to project I, then the resulting net profit is N(i,j). The problem is to allocate the resource to the r projects is such a way as to maximize total net profit. This problem can be formulated as r+1 stage graph problem.

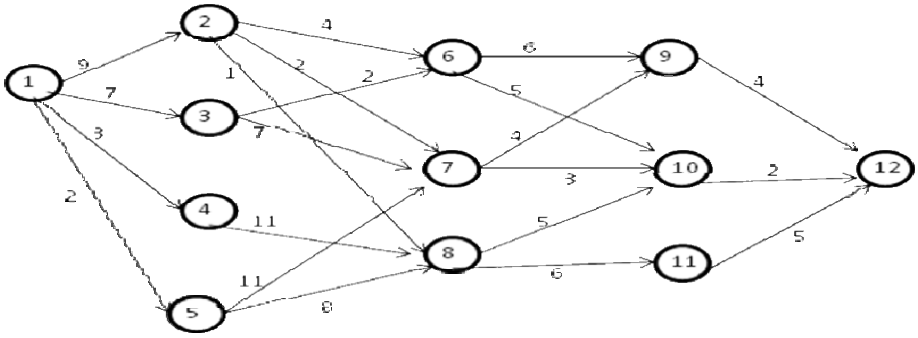
A dynamic programming formulation for a k-stage graph problem is obtained by first noticing that every S to T path is the result of k-2 decisions. The i^{th} decision involves in determining which vertex v_{i+1} where 1<=i<=k-2 is to be on the path. Let p(i,j) be a minimum cost path from vertex 'j' in v_i to vertex 'T'. Let cost(i,j) be the cost of this path. This multistage graph problem can be solved using 2 approaches.

1) Forward approach 2) Backward approach

Forward Approach:

$$cost(i,j) = \min_{\substack{l \epsilon V_{i+1} \\ <j,l> \epsilon E}} \{ c(j,l) + cost(i+1,l) \}$$

STAGES **V_1** **V_2** **V_3** **V_4** **V_5**



Five-Stage Graph

First compute $\cos t[k-2, j] \forall j \in v_{k-2}$ and then compute $\cos t[k-3, j] \forall j \in v_{k-3}$ and so on, and finally compute cost[1,S]

There are 5 stages in the graph V1 V2 V3 V4 V5

Stage 4 contains 3 vertices (9,10,11) cost of each vertex at stage 4 to the destination

cost(4,9) = c(9,12) = 4 stage 4 cost from vertex 9 to vertex 12
cost(4,10) = c(10,12) = 2 stage 4 cost from vertex 10 to vertex 12
cost(4,11) = c(11,12) = 5 stage 4 cost from vertex 11 to vertex 12

Stage 1: $\cos t[k-2,\ j] \forall j \in v_{k-2}$

$\cos t[3,\ j] \forall\ j \in v_3 = \{6, 7, 8\}$

cost(3,6) means that 3 refers to stage. At stage 3 cost for vertex 6 to destination

cost(3,6) =min { c(6,9) + cost(4,9) = 6 + 4 =10 ; c(6,10) + cost(4,10) = 5 + 2 = 7 } 7

cost(3,7) =min { c(7,9) + cost(4,9) = 4 + 4 = 8 ; c(7,10) + cost(4,10) = 3 + 2 = 5 } 5

cost(3,8) = min { c(8,10) + cost(4,10) = 5 + 2 =7 ; c(8,11) + cost(4,11) = 6 + 5 = 11 } 7

cost(2,2) =min { 4 + cost(3,6) ; 2 + cost(3,7) ; 1 + cost(3,8) } 7

cost(2,3) = 9

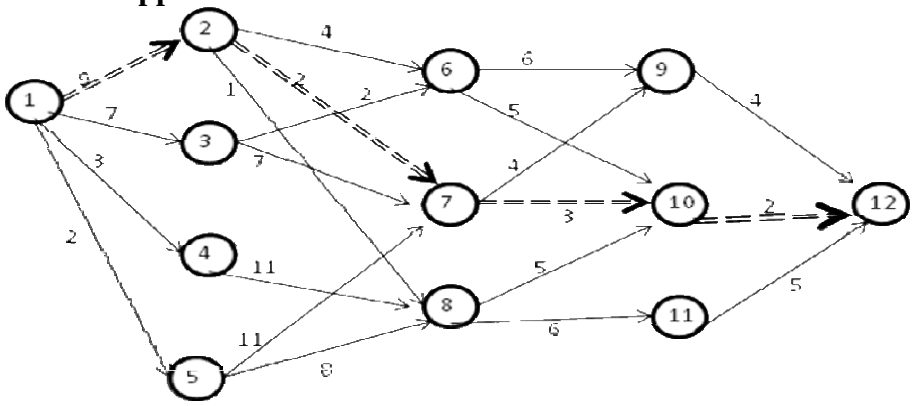
cost (2,4)=18

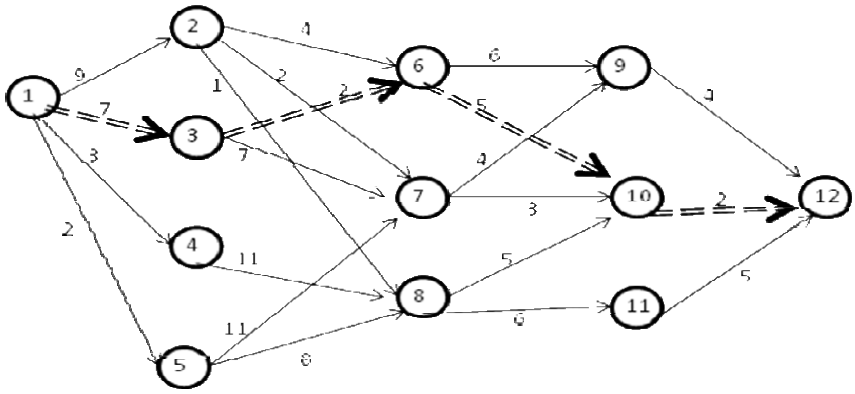
cost(2,5)=15

cost(1,1) = min { 9+cost(2,2) ; 7+cost(2,3) ; 3+cost(2,4) ; 2+cost(2,5) } 16

Note that in the calculation of cost(2,2), we have reused the values of cost (3,6), cost (3,7) and cost(3,8) and so avoided their recomputation. A minimum cost s to t path has a cost of 16. This path can be determined easily if we record the decision made at each stage (vertex).

Backward approach:





5) TRAVELLING SALES PERSON

The tour of graph G is a directed simple cycle that includes every vertex in V. The cost of the tour is the sum of cost of the edges on the tour. The travelling sales-person problem is to find a tour of minimum cost.

Applications: Suppose we have to route a postal van to pick up mail from mailbox located at 'n' different sites. An (n+1) vertex graph can be used to represent the situation. One vertex represents the post office from which the postal van starts and to which it must return. Edge <i,j> is assigned a cost equal to the distance from site 'I' to site 'j'. Route taken by the postal van is a tour and we are interested in finding tour of minimum length.

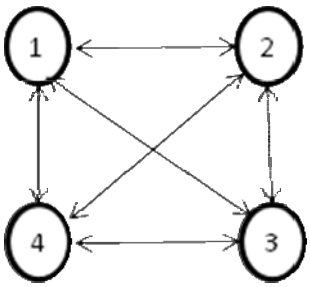
Every tour consist of an edge <1, k> for some k $\in$ V-{1} and a path from vertex k to vertex 1, which goes through each vertex in V-{1,k} exactly once. It is easy to see that if the tour is optimal, then the path from k to 1 must be a shortest k to 1 path going through all vertices in V-{1, k}. Hence, the principle of optimality holds.

Let g(i,S) be the length of a shortest path starting at vertex 'i' going through all vertices in 'S' & terminating at vertex 1. The function g(1,V-{1}) is the length of an optimal salesperson tour. From the principle of optimality

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

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

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



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

Find g(1,{2,3,4}) V{1,2,3,4}

Solution:

Stage 1) compute **g(i,S)** where |S| = ø, **with no intermediate nodes**.

i $\in$ {v-{1} } $i \neq 1, 1 \notin S$ *and* $i \notin S$

i={2,3,4}
g(2,ø)=c_{21}=5
g(3,ø)=c_{31}=6
g(4,ø)=c_{41}=8

Stage 2) Compute **g(i,S)** where |S|=1, **with one intermediate node**

i$\in${v-{1}} $i \neq 1, 1 \notin S$ *and* $i \notin S$

i={2,3,4} s={2/3/4} & <i,s> should be an edge (j=s)

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

g(2,{3}) = c23+g(3,s-{3}) = 9 + g(3,ø) = 9 + c31 = 9+6 = 15
g(2,{4}) = c24+g(4,s-{4}) = 10 + g(4,ø) = 10 + c41 = 10+8=18
g(3,{2}) = c32+g(2,s-{2}) = 13 + g(2,ø) = 13 + c21 = 13+5=18
g(3,{4}) = c34+g(4,s-{4}) = 12 + g(4,ø) = 12 + c41 = 12+8=20
g(4,{2}) = c42+g(2,s-{2}) = 8 + g(2,ø) = 10 + c21 = 8 + 5=13
g(4,{3}) = c43+g(3,s-{3}) = 9 + g(3,ø) = 9 + c31 = 9 + 6=15

Step 3) Compute **g(i,s)** where |S|=2, **with two intermediate nodes**

$i \in (v-\{2\}) \qquad i \neq 1, 1 \notin S \text{ and } i \notin S$

i={2,3,4}, S={{2,3}/{3,4}/{2,4}}

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

$$g(2,\{3,4\}) = \min \begin{cases} c_{23} + g(3,s-\{3\}) & \text{when j=3} \\ c_{24} + g(4,s-\{4\}) & \text{when j=4} \end{cases} \qquad j=\{3,4\}$$

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

$$g(3,\{2,4\}) = \min \begin{cases} c_{32} + g(2,s-\{2\}) \\ c_{34} + g(4,s-\{4\}) \end{cases} \qquad j=\{2,4\}$$

$$\min \begin{cases} 13 + g(2,\{4\}) \\ 12 + g(4,\{2\}) \end{cases} = \min \begin{cases} 13+18 \\ 12+13 \end{cases} = 25$$

$$g(4,\{2,3\}) = \min \begin{cases} c_{42} + g(2,s-\{2\}) \\ c_{43} + g(3,s-\{3\}) \end{cases} \qquad j=\{2,3\}$$

$$\min \begin{cases} 8 + g(2,\{3\}) \\ 9 + g(3,\{2\}) \end{cases} = \min \begin{cases} 8+15 \\ 9+18 \end{cases} = \min\{23,27\} = 23$$

Stage 4) Compute **g(i,S)** where |S|=3, **with three intermediate nodes**

$i \in \{v-\{1\}$ and $i=\{1\}$,

S={2,3,4}

j={2/3/4}

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

$$g(1,\{2,3,4\}) = \min \begin{cases} C_{12} + g(2,S-\{2\}) & \text{when j=2} \\ C_{13} + g(3,S-\{3\}) & \text{when j=3} \\ C_{14} + g(4,S-\{4\}) & \text{when j=4} \end{cases}$$

$$\min \begin{cases} 10 + g(2,\{3,4\}) \\ 15 + g(3,\{2,4\}) \\ 20 + g(4,\{2,3\}) \end{cases} = \min \begin{cases} 10 + 25 \\ 15 + 25 \\ 20 + 23 \end{cases} \min\{35,40,43\} = 35$$

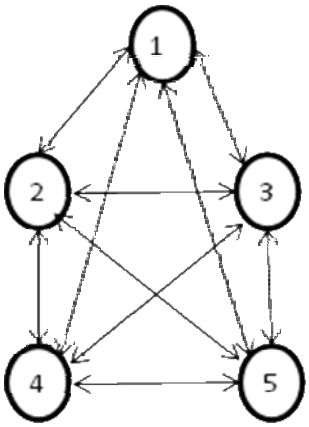
Optimal cost of tour is 35

$g(1,\{2,3,4\}) = \{C_{12} + g(2,\{3,4\})\}$ thus the tour starts form 1 and goes to 2
$g(2,\{3,4\}) = \{C_{24} + g(4,\{3\})\}$ then from 2 to 4
$g(4,\{3\}) = \{C_{43} + g(3,\{\Phi\})\}$ then from 4 to 3
$g(3,\{\Phi\}) = C_{31}$ and from 3 to 1

$C_{12}\ C_{24}\ C_{43}\ C_{31}$

The sequence of tour is 1 2 4 3 1

Example 2) Travelling Salesperson



	1	2	3	4	5
1	0	2	1	2	1
2	1	0	3	2	5
3	6	2	0	2	1
4	1	1	2	0	2
5	3	1	2	1	0

Compute g(1,{2,3,4,5}) V={1,2,3,4,5}

Stage 1) Compute g(i,s) where |s|=ø , i ϵ {v- { s }}
i={2,3,4,5} with no intermediate nodes

g (2,ø)=C_{21} =1
g (3,ø)=C_{31} =6
g (4,ø)=C_{41} =1
g (5,ø)=C_{51} =3

Stage 2) Compute g(i,s) where |s|=1 , i ϵ {v- { s }}
With one intermediate node
i={2,3,4,5} s = {2,3,4,5} & <i,s> should be an edge

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

g(2,{3}) = C_{23} + g(3,s-{3}) = 3 + g(3,ø) = 3 + 6 = 9
g(2,{4}) = C_{24} + g(4,s-{4}) = 2 + g(4,ø) = 2 + 1 = 3
g(2,{5}) = C_{25} + g(5,s-{5}) = 5 + g(5,ø) = 5 + 3 = 8

g(3,{1}) = C_{31} + g(1,s-{1}) = 6 + g(1,ø) = 6 + 0 = 6
g(3,{2}) = C_{32} + g(2,s-{2}) = 2 + g(2,ø) = 2 + 1 = 3
g(3,{4}) = C_{34} + g(4,s-{4}) = 2 + g(4,ø) = 2 + 1 = 3
g(3,{5}) = C_{35} + g(5,s-{5}) = 1 + g(5,ø) = 1 + 3 = 4

g(4,{1}) = C_{41} + g(1,s-{1}) = 1 + g(1,ø) = 1 + 0 = 1
g(4,{2}) = C_{42} + g(2,s-{2}) = 1 + g(2,ø) = 1 + 1 = 2
g(4,{3}) = C_{43} + g(3,s-{3}) = 2 + g(3,ø) = 2 + 6 = 8
g(4,{5}) = C_{45} + g(5,s-{5}) = 2 + g(5,ø) = 2 + 3 = 5

g(5,{1}) = C_{51} + g(1,s-{1}) = 3 + g(1,ø) = 3 + 0 = 3
g(5,{2}) = C_{52} + g(2,s-{2}) = 1 + g(2,ø) = 1 + 1 = 2
g(5,{3}) = C_{53} + g(3,s-{3}) = 2 + g(3,ø) = 2 + 6 = 8
g(5,{4}) = C_{54} + g(4,s-{4}) = 1 + g(4,ø) = 1 + 1 = 2

Stage 3) Compute g(i,s) where |s|=2 ,i.e i ϵ {v- { s }}

With two intermediate nodes

i={2,3,4,5} s = {2,3,4,5}

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

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

$$\min\begin{Bmatrix} 3 + g(3,\{4\}) \\ 2 + g(4,\{3\}) \end{Bmatrix} = \min\begin{Bmatrix} 3 + 3 \\ 2 + 8 \end{Bmatrix} = \min\begin{Bmatrix} 6 \\ 10 \end{Bmatrix} = 6$$

$$g(2,\{4,5\}) = \min\begin{Bmatrix} C_{24} + g(4,S-\{4\}) \\ C_{25} + g(5,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 2 + g(4,\{5\}) \\ 5 + g(5,\{4\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 5 \\ 5 + 2 \end{Bmatrix} = \min\begin{Bmatrix} 7 \\ 7 \end{Bmatrix} = 7$$

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

$$\min\begin{Bmatrix} 2 + g(2,\{4\}) \\ 2 + g(4,\{2\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 3 \\ 2 + 2 \end{Bmatrix} = \min\begin{Bmatrix} 5 \\ 4 \end{Bmatrix} = 4$$

$$g(3,\{2,5\}) = \min\begin{Bmatrix} C_{32} + g(2,S-\{2\}) \\ C_{35} + g(5,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 2 + g(2,\{5\}) \\ 1 + g(5,\{2\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 8 \\ 1 + 2 \end{Bmatrix} = \min\begin{Bmatrix} 10 \\ 3 \end{Bmatrix} = 3$$

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

$$\min\begin{Bmatrix} 1 + g(2,\{3\}) \\ 2 + g(3,\{2\}) \end{Bmatrix} = \min\begin{Bmatrix} 1 + 9 \\ 2 + 3 \end{Bmatrix} = \min\begin{Bmatrix} 10 \\ 5 \end{Bmatrix} = 5$$

$$g(4,\{2,5\}) = \min\begin{Bmatrix} C_{42} + g(4,S-\{2\}) \\ C_{45} + g(4,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 1 + g(4,\{5\}) \\ 2 + g(5,\{2\}) \end{Bmatrix} = \min\begin{Bmatrix} 1 + 5 \\ 2 + 2 \end{Bmatrix} = \min\begin{Bmatrix} 6 \\ 4 \end{Bmatrix} = 4$$

$$g(5,\{2,3\}) = \min\begin{Bmatrix} C_{52} + g(2,S-\{2\}) \\ C_{53} + g(3,S-\{3\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 1 + g(2,\{3\}) \\ 2 + g(3,\{2\}) \end{Bmatrix} = \min\begin{Bmatrix} 1 + 9 \\ 2 + 3 \end{Bmatrix} = \min\begin{Bmatrix} 10 \\ 5 \end{Bmatrix} = 5$$

$$g(5,\{2,4\}) = \min\begin{Bmatrix} C_{52} + g(2,S-\{2\}) \\ C_{54} + g(4,S-\{4\}) \end{Bmatrix}$$

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

$$g(2,\{3,5\}) = \min\begin{Bmatrix} C_{23} + g(3,S-\{3\}) \\ C_{25} + g(5,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 3 + g(3,\{5\}) \\ 5 + g(5,\{3\}) \end{Bmatrix} = \min\begin{Bmatrix} 3 + 4 \\ 5 + 8 \end{Bmatrix} = \min\begin{Bmatrix} 7 \\ 13 \end{Bmatrix} = 7$$

$$g(3,\{4,5\}) = \min\begin{Bmatrix} C_{34} + g(4,S-\{4\}) \\ C_{35} + g(5,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 2 + g(4,\{5\}) \\ 1 + g(5,\{4\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 5 \\ 1 + 2 \end{Bmatrix} = \min\begin{Bmatrix} 7 \\ 3 \end{Bmatrix} = 3$$

`

$$g(4,\{3,5\}) = \min\begin{Bmatrix} C_{43} + g(3,S-\{3\}) \\ C_{45} + g(5,S-\{5\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 2 + g(3,\{5\}) \\ 2 + g(5,\{3\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 4 \\ 2 + 8 \end{Bmatrix} = \min\begin{Bmatrix} 6 \\ 10 \end{Bmatrix} = 6$$

$$g(5,\{3,4\}) = \min\begin{Bmatrix} C_{53} + g(3,S-\{3\}) \\ C_{54} + g(4,S-\{4\}) \end{Bmatrix}$$

$$\min\begin{Bmatrix} 2 + g(3,\{4\}) \\ 1 + g(4,\{3\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 3 \\ 1 + 8 \end{Bmatrix} = \min\begin{Bmatrix} 5 \\ 9 \end{Bmatrix} = 5$$

Stage 4) Compute g(i,s) where |s|=3 ,i.e i $\in$ {v- { s }}

With three intermediate nodes

i={2,3,4,5} s = {2,3,4,5}

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

$$g(2,\{3,4,5\}) = \min\begin{Bmatrix} C_{23} + g(3,S-\{3\}) \\ C_{24} + g(4,S-\{4\}) \\ C_{25} + g(5,S-\{5\}) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 3 + g(3,\{4,5\}) \\ 2 + g(4,\{3,5\}) \\ 5 + g(5,\{3,4\}) \end{Bmatrix} = \min\begin{Bmatrix} 3 + 3 \\ 2 + 6 \\ 5 + 5 \end{Bmatrix} = \min\begin{Bmatrix} 6 \\ 8 \\ 10 \end{Bmatrix} = 6$$

$$g(3,\{2,4,5\}) = \min\begin{Bmatrix} C_{32} + g(2,S-\{2\}) \\ C_{34} + g(4,S-\{4\}) \\ C_{35} + g(5,S-\{5\}) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 2 + g(2,\{4,5\}) \\ 2 + g(4,\{2,5\}) \\ 1 + g(5,\{2,4\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 7 \\ 2 + 4 \\ 1 + 3 \end{Bmatrix} = \min\begin{Bmatrix} 9 \\ 6 \\ 4 \end{Bmatrix} = 4$$

$$g(4,\{2,3,5\}) = \min\begin{Bmatrix} C_{42} + g(2,S-\{2\}) \\ C_{43} + g(3,S-\{3\}) \\ C_{45} + g(5,S-\{5\}) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 1 + g(2,\{3,5\}) \\ 2 + g(3,\{2,5\}) \\ 2 + g(5,\{2,3\}) \end{Bmatrix} = \min\begin{Bmatrix} 1 + 7 \\ 2 + 3 \\ 2 + 5 \end{Bmatrix} = \min\begin{Bmatrix} 8 \\ 5 \\ 7 \end{Bmatrix} = 5$$

$$g(5,\{2,3,4\}) = \min\begin{Bmatrix} C_{52} + g(2,S-\{2\}) \\ C_{53} + g(3,S-\{3\}) \\ C_{54} + g(4,S-\{4\}) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 1 + g(2,\{3,4\}) \\ 2 + g(3,\{2,4\}) \\ 1 + g(4,\{2,3\}) \end{Bmatrix} = \min\begin{Bmatrix} 1 + 6 \\ 2 + 4 \\ 1 + 5 \end{Bmatrix} = \min\begin{Bmatrix} 7 \\ 6 \\ 6 \end{Bmatrix} = 6$$

Stage 5) Compute g(i,s) where |s|=4 ,i.e i $\in$ {v- { 1 }}
With 4 intermediate nodes

i={1} s = {2,3,4,5}

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

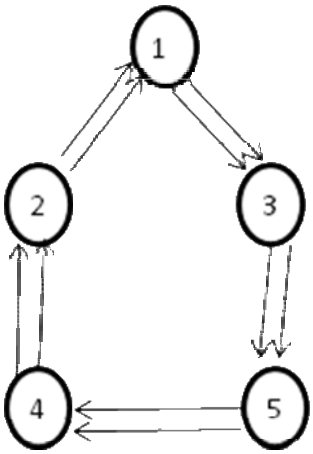
$$g(1,\{2,3,4,5\}) = \min\begin{Bmatrix} C_{12} + g(2,S-\{2\}) \\ C_{13} + g(3,S-\{3\}) \\ C_{14} + g(4,S-\{4\}) \\ C_{15} + g(5,S-\{5\}) \end{Bmatrix}$$

$$= \min\begin{Bmatrix} 2 + g(2,\{3,4,5\}) \\ 1 + g(3,\{2,4,5\}) \\ 2 + g(4,\{2,3,5\}) \\ 1 + g(5,\{2,3,4\}) \end{Bmatrix} = \min\begin{Bmatrix} 2 + 6 \\ 1 + 4 \\ 2 + 5 \\ 1 + 6 \end{Bmatrix} = \min\begin{Bmatrix} 8 \\ 5 \\ 7 \\ 7 \end{Bmatrix} = 5$$

Optimal Cost tour is 5

$g(1,\{2,3,4,5\}) = C_{13} + g(3,\{2,4,5\})$ thus tour stars from 1 and goes to 3
$g(3,\{2,4,5\} = C_{35} + g(5,\{2,4\})$ then from 3 to 5
$g(5,\{2,4\}) = C_{52} + g(2,\{4\})$ then from 5 to 2
$g(2,\{4\}) = C_{24} + g(4,\{ø\})$ then from 2 to 4
$g(4,ø) = C_{41}$ then from 4 to 1

Optimal tour is 1-3-5-4-2-1



6) All pairs shortest path problem

Let G=(V,E) be a directed graph consisting of n vertices and each edge is associated with a weight. The problem of finding the shortest path between all pairs of vertices in a graph is called all pairs shortest path problem. This problem can be solved by using Dynamic programming Technique.

The all pair shortest path problem is to determine a matrix A such that A(i,j) is the length of a shortest path from vertex i to j. Assume that this path contains no cycles. If k is an intermediate vertex on this path, then the sub paths form i to k and from k to j are the shortest paths from I to k and from k to j respectively.

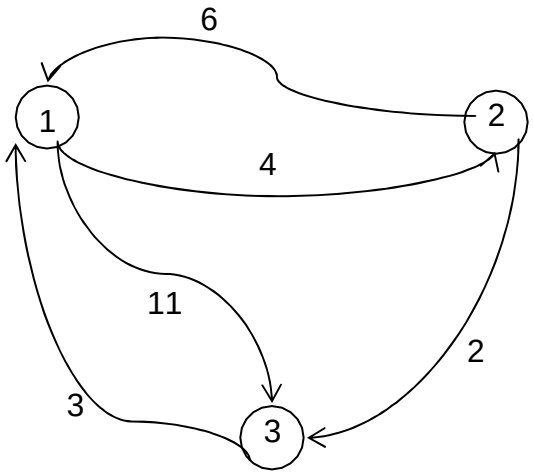
Otherwise the path from i to j is not shortest path. If k is intermediate vertex with highest index then the path i to k is the shortest path going through no vertex with index greater than k-1. similarly the path k to j is shortest path going through no vertex with index greater than k-1.

The shortes path can be computed using following recursive method.

$$A^k(i,j) = W(i,j) \quad if \quad k = 0$$

$$A^k(i,j) = min\{A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j)\} \quad if \quad k \geq 1$$

Example : Compute all pairs shortest path for the following graph



Graph G

$$cost\ Adjacency\ Matrix\ A^0(i,j) = W(i,j) = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

Step 1

For k=1 i.e. going from i to j through intermediate vertex 1

When i=1 j=1/2/3

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

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

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

When i=2 j=1/2/3

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

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

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

When i=3 j=1/2/3

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

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

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

$$\text{cost Adjacency Matrix } A^1(i,j)=\begin{bmatrix}0 & 4 & 11\\ 6 & 0 & 2\\ 3 & 7 & 0\end{bmatrix}$$

Step 2

For k=2 i.e. going from i to j through intermediate vertex 2

When i=1 k=2 j=1/2/3

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

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

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

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

When i=2 j=1/2/3

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

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

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

When i=3 j=1/2/3

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

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

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

$$cost\ Adjacency\ Matrix\ A^2(i,j) = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Step 3

For k=3 i.e. going from i to j through intermediate vertex 3

When i=1 k=3 j=1/2/3

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

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

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

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

When i=2 j=1/2/3

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

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

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

When i=3 j=1/2/3

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

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

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

$$cost\ Adjacency\ Matrix\ A^3(i,j) = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

This matrix gives the all pairs shortest path solution

Algorithm all_pairs_shortest_path(W,A,n)

```
// W is weighted array matrix, n is the number of vertices,
// A is the cost of shortest path from vertex i to j.
{
   for i:= 1 to n do
      for j:= 1 to n do
         A[i,j]:= W[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]
}
```

The Time Complexity for this method is $O(n^3)$