Dynamic Programming

Syllabus: General Method, Applications- Matrix Chain Multiplication, Optimal Binary Search Trees, 0/1 Knapsack Problem, All pairs Shortest Path Problem, Travelling Sales person Problem, Reliability Design.

General Method:

Dynamic Programming is useful for solving multistage optimization problems, especially sequential decision problems. Here the word "dynamic" refers to some sort of time reference & "Programming" is interpreted as planning or tabulation rather than programming that is encountered in computer programs.

Dynamic Programming employs the following steps as shown below:

Step-1: The given problem is divided into a number of subproblem as in "Divide and Conquer" strategy. But in divide and conquer, the subproblems are independent of each other but in dynamic programming, there are all overlapping subproblems. A recursive formulatio is formed of the given problem.

Step-2: The problem, usually solved in bottom-up manner. To avoid, repeated computation of multiple overlapping subproblems, a table is created. When a subproblem is solved, then its solution is stored in the table. Then the solutions are combined to solve the overall problem.

Applications:

1. Matrix chain Multiplication
2. Optimal Binary Search Trees.
3. 0/1 Knapsack Problem
4. All Pairs Shortest path Problem
5. Travelling Sales Person Problem
6. Realiability Design.

Divide & Conquer

1. Divide & conquer is a problem solving approach.
2. In this approach, we divide the problem into subproblems, solve the individual subproblems & then combine the subproblems to get the solution to the main problem.
3. Divide and conquer splits its i/p at specific deterministic points usually in the middle.
4. Divide & conquer uses top down approach while solving any problem.
5. Divide & conquer is recursive
6. Applications are Binary search, Quick sort, Merge sort etc.

Greedy Method

1. Greedy Method is used for obtaining optimum solution.
2. The Greedy Method is a straight forward approach for choosing the optimum choice or solution.
3. The i/p's are considered in an order determined by some selection procedure.
4. Greedy Method uses top down approach while solving any problem.
5. Greedy Method is not recursive
6. Applications are Minimum cost, Spanning Trees, Knapsack problem etc

Dynamic programming

1. Dynamic Programming is also used for obtaining optimum solution
2. Dynamic Programming considers all possible sequences in order to obtain the optimum solution.
3. Dynamic programming splits its i/p at every possible split points rather than at a particular point
4. Dynamic programming uses bottom up approach while solving any problem.
5. Dynamic programming is not recursive
6. Applications are 0/1 knapsack problem, Matrix Chain multiplication etc.

Matrix chain Multiplication:

Input: n matrices $A_1, A_2 \ldots A_n$ of dimensions $P_1 \times P_2, P_2 \times P_3 \ldots$ $P_n \times P_{n+1}$

Problem: In what order they should be multiplied so that it would take minimum number of computations to derive the product.

Problem: Goal: To compute the matrix product $A_1, A_2 \ldots A_n$.

Consider an eg, of the best way of multiplying 3 matrices. Let A_1 of dimensions 5×4, A_2 of dimensions 4×6 & A_3 of dimensions 6×2

$A_1 * (A_2 \times A_3)$ takes $(5 \times 4 \times 2) + (4 \times 6 \times 2) = 88$

$(A_1 \times A_2) \times A_3$ takes $(5 \times 4 \times 6) + (5 \times 6 \times 2) = 180$.

Thus $A_1 * (A_2 \times A_3)$ is much cheaper to compute that $(A_1 \times A_2) \times A_3$, although both lead to the same result. Hence optimal cost is 88.

To solve this problem using dynamic programming we will perform following steps:

Step-1: Let M_{ij} denote the cost of multiplying $A_i \ldots A_j$, where the cost is measured in the no. of scalar multiplications.

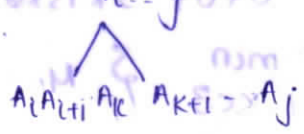
$M[i,i] = 0 \quad \forall i$

$M[i,j] = \infty \quad 0 < j$.

Step-2: Let T be the tree corresponding to the optimal way of multiplying $A_i \ldots A_j$.

T has left subtree L & a right subtree R.

L corresponds to multiplying $A_i \ldots A_K$ & R to multiplying $A_{K+1} \ldots A_j$ for some integer K



Step-3: We will apply formula for computing each sequence.

$$M_{ij} = \min \{ M_{iK} + M_{K+1,j} + P_i P_{K+1} P_{j+1} \mid i \leq K < j \}$$

Eg-1 Consider $A1 = 30 \times 40$ $A2 = 40 \times 5$, $A3 = 5 \times 15$, $A4 = 15 \times 6$

$P1 = 30$ $P2 = 40$ $P3 = 5$ $P4 = 15$ $P5 = 6$

Always $M_{ii} = 0$

$M_{ij} = \infty$ for $i > j$

	1	2	3	4
1	0	6000	8250	7350
2	∞	0	3000	1650
3	∞	∞	0	450
4	∞	∞	∞	0

$$M_{ij} = \min_{i \le k < j} \{ M_{ik} + M_{k+1,j} + P_i P_{k+1} P_{j+1} \}$$

$$M_{12} = \min_{1 \le k < 2} \{ M_{11} + M_{12} + P_1 P_2 P_3 \}$$

$\Rightarrow k = 1$

$= 0 + 0 + (30 \times 40 \times 5) = 6000$

$$M_{23} = \min_{2 \le k < 3} \{ M_{22} + M_{33} + P_2 P_3 P_4 \}$$

$\Rightarrow k = 2$

$= 0 + 0 + 40 \times 5 \times 15$

$= 3000$

$$M_{34} = \min_{3 \le k < 4} \{ M_{33} + M_{44} + P_3 P_4 P_5 \}$$

$\Rightarrow k = 3$

$= 0 + 0 + 5 \times 15 \times 6 = 450.$

$$M_{13} = \min_{1 \le k < 3} \{ M_{11} + M_{33} + P_1 P_2 P_4, \; M_{12} + M_{33} + P_1 P_3 P_4 \}$$

$\Rightarrow k = 1, 2$

$= \{ 0 + 3000 + 30 \times 40 \times 15, \; 6000 + 0 + 30 \times 5 \times 15 \}$

$= \min \{ 21000, 8250 \}$

$= 8250.$

$$M_{24} = \min_{i \le k < j} \{ M_{ik} + M_{k+1,j} + P_i P_{k+1} P_{j+1} \}$$

$$= \min_{2 \le k < 4} \Rightarrow k = 2, 3$$

$$= \min_{2 \le k < 4} \{ M_{22} + M_{34} + P_2 P_3 P_5 , \; M_{23} + M_{44} + P_2 P_4 P_5 \}$$

$$= \min_{2 \le k < 4} \{ 0 + 450 + (40 \times 5 \times 6) , \; 3000 + 0 + (40 \times 15 \times 6) \}$$

$$= \min \{ 1650, 6600 \}$$

$$= 1650.$$

$$M_{14} = \min_{1 \le k < 4} \Rightarrow k = 1, 2, 3.$$

$$= \min_{1 \le k < 4} \{ M_{ik} + M_{k+1,j} + P_i P_{k+1} P_{j+1} \}$$

$$= \min_{1 \le k < 4} \{ M_{11} + M_{24} + P_1 P_2 P_5 , \; M_{12} + M_{34} + P_1 P_3 P_5 , \; M_{13} + M_{44} + P_1 P_4 P_5 \}$$

$$= \min \{ 0 + 1650 + (30 \times 40 \times 6), \; 6000 + 450 + (30 \times 5 \times 6), \; 8250 + 0 + (30 \times 15 \times 6) \}$$

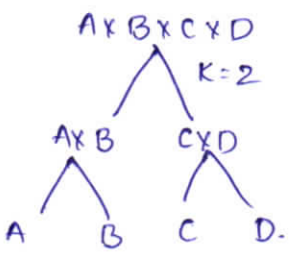
$$= \min \{ 8850, 7350, 10950 \}$$

$$= 7350.$$

Now the table becomes,

	1	2	3	4
1	0	6000	8250	7350
2	x	0	3000	1650
3	x	x	0	450
4	x	x	x	0

As we get minimum value of M_{14} when $k = 2$



∴ The optimum sequence is $(A \times B) \times (C \times D)$.

Optimum cost = 7350

For instance,

$$(A \times B) \times (C \times D) = (30 \times 40 \times 5) + (5 \times 15 \times 6) + (30 \times 5 \times 6)$$

$$= 6000 + 450 + 900$$

$$= 7350$$

$A \times B = 30 \times 40$, $B \times C = 40 \times 5$ } 30×5

$C = 5 \times 15$, $D = 15 \times 6$ } 5×6

Algorithm:

There are two algorithms, one for computing the matrix chain multiplication for optimum valued multiplication sequence & another is to extract the optimum sequence.

```
Algorithm Matrixchain-Mul(array P[1...n])
{
  for i:=1 to n do
    M[i,i] := 0;
  for len:=2 to n do
  {
    for i:=1 to (n-len+1) do
    {
      j := i+len-1;
      M[i,j] := ∞;
      for k:=i to j-1 do
      {
        q := M[i,k] + M[k+1,j] + P[i] * P[k] * P[j];
        if (q < M[i,j])
          M[i,j] := q;
      }
      S[i,q] := k;
  }
}
```

For extracting optimum sequence,

```
Algorithm MUL(i,j)
{
  if (i==j) then
    return A[i];
  else
  {
    K := S[i,j];
    P := MUL(i,K);
    Q := MUL(K+1,j);
  }
  return P*Q;
}
```

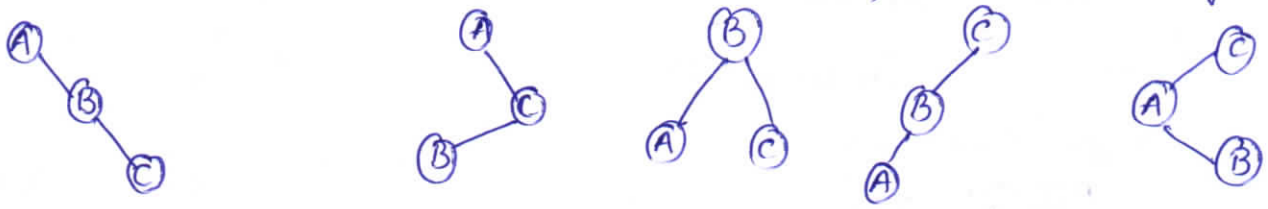
Analysis:

Using three nested for loops each M[i,j] called many times This leads to $O(n^3)$ time complexity.

Optimal Binary Search Trees:

Given a fixed set of identifiers, we wish to create a binary search tree organization. We may expect different binary search trees for the same identifier set to have different performance characteristics.

Suppose there are three nodes or three keys (A, B, C). To arrange these nodes in a BST we will have $5\left(^{2n}C_n\left(\frac{1}{n+1}\right)\right)$ possible arrangements like,



(i) (ii) (iii) (iv) (v)

Out of which, (iii) is most optimum because in this BST, any node can be accessed more efficiently than any other arrangement.

Now with this approach only optimal Binary search tree is invented. The element having more probability of appearance should be placed nearer to the root of the Binary Search Tree & the element which appears for least no of times is placed away from the root. The BST created with such kind of arrangement is called optimal Binary Search Tree.

Let $\{a_1, a_2 \ldots a_n\}$ be the set of identifiers such that $a_1 < a_2 < \ldots a_n$. Let, P(i) be the probability with which we can search for a_i → Successful Search.

Let q_i be the probability of searching an element x such that $a_i < x < a_{i+1}$ where $0 \le i \le n$ → unsuccessful search. Thus P(i) is Probability of successful search & q(i) is the probability of unsuccessful search. Then a tree which is build with optimum cost from $\sum_{i=1}^{n} P(i)$ & $\sum_{i=1}^{n} q(i)$ is called Optimal Binary Search Tree.

To solve the problem of OBST, we use dynamic programming approach.

1. Characterizing the optimal substructure of an OBST
2. A recursive solution:

To obtain a recursive solution, we have to compute, weight of each tree as sum of q_i's & p_j probabilities in the subtree. The weight can be denoted by

$$\boxed{w_{ij} = w_{ij-1} + p_j + q_j}$$ always $w_{ii} = q_i$, $c_{ii} = 0$, $r_{ii} = 0$

Cost can be computed using

$$\boxed{c_{ij} = \min_{i<k\le j} \{ c_{ik-1} + c_{kj} + w_{ij} \}}$$

Root is

$$\boxed{r_{ij} = k}$$

3. Computing the cost of OBST & construct OBST.

Eg: Consider $\{a_1, a_2, a_3, a_4\} = \{$do, if, int, while$\}$

$P[1,2,3,4] = \{3, 3, 1, 1\}$

$q[0,1,2,3,4] = \{2, 3, 1, 1, 1\}$

Sol: To construct the optimal Binary Search Tree, we will apply following formulae for computations of w, c & r.

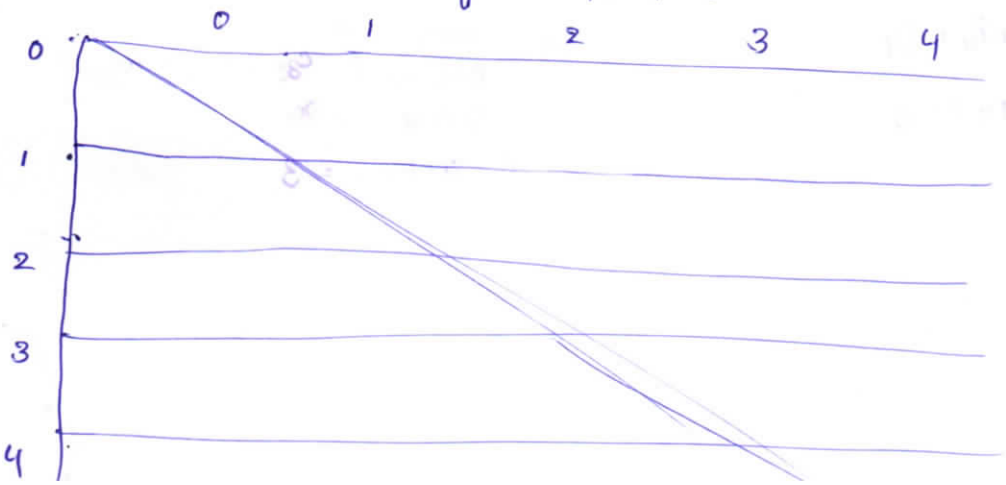
$w_{ii} = q_i \quad r_{ii} = 0 \quad c_{ii} = 0$

$w_{ij} = w_{ij-1} + p_j + q_j$

$c_{ij} = \min_{i<k\le j} \{ c_{ik-1} + c_{kj} + w_{ij} \}$

$r_{ij} = k$

we will construct tables for w, c & r.



	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	$w_{04}=16$ $c_{04}=32$ $r_{04}=2$				

$w_{ij} = w_{ij-1} + p_j + q_j$

$w_{01} = w_{00} + p_1 + q_1$

$= 2+3+3$

$= 8$

$c_{ij} = \min_{i<k\le j} \{ c_{ik-1} + c_{kj} + w_{ij} \}$

$c_{01} = \min_{0<k\le 1} \{ c_{00} + c_{01} + w_{01} \}$

$\Rightarrow k = 1$

$= 0+0+8 = 8$

$r_{ij} = k$

$r_{01} = 1$

$w_{12} = w_{11} + p_2 + q_2$

$= 3+3+31$

$= 7$

$c_{12} = \min_{1<k\le 2} \{ c_{11} + c_{22} + w_{12} \}$

$\Rightarrow k = 2$

$= 0+0+7 = 7$

$r_{ij} = k$

$r_{12} = 2$

$w_{23} = w_{22} + p_3 + q_3$

$= 1+1+1 = 3$

$c_{23} = \min_{2<k\le 3} \{ c_{22} + c_{33} + w_{23} \}$

$\Rightarrow k = 3$

$= 0+0+3 = 3$

$r_{ij} = k$

$r_{23} = 3$

$w_{34} = w_{33} + p_4 + q_4$

$= 1+1+1 = 3$

$c_{34} = \min_{3<k\le 4} \{ c_{33} + c_{44} + w_{34} \}$

$\Rightarrow k = 4$

$= 0+0+3 = 3$

$r_{ij} = k$

$r_{34} = 4.$

$w_{ij} = w_{ij-1} + p_j + q_j$

$w_{02} = w_{01} + p_2 + q_2$

$= 8 + 3 + 1$

$= 12$

$c_{ij} = \min_{i<k\le j} \{ c_{ik-1} + c_{kj} + w_{ij} \}$ $\quad r_{ij} = k$

$c_{02} = \min_{0<k\le 2} \{ c_{00} + c_{12} + w_{02},\ c_{01} + c_{22} + w_{02} \}$ $\quad r_{02} = 1$

$\Rightarrow k = 1, 2$

$= \min_{0<k\le 2} \{ 0 + 7 + 12,\ 8 + 0 + 12 \}$

$= \min \{ 19, 20 \}$

$= 19$ (for $k = 1$)

$w_{13} = w_{12} + p_3 + q_3$

$= 7 + 1 + 1 = 9$

$c_{13} = \min_{1<k\le 3} \{ c_{11} + c_{23} + w_{13},\ c_{12} + c_{33} + w_{13} \}$ $\quad r_{ij} = 2$

$\Rightarrow k = 2, 3$

$= \min \{ 0 + 3 + 9,\ 7 + 0 + 9 \}$

$= \min \{ 12, 16 \}$

$= 12$ (for $k = 2$)

$w_{24} = w_{23} + p_4 + q_4$

$= 3 + 1 + 1 = 5$

$= 5$

$c_{24} = \min_{2<k\le 4} \{ c_{22} + c_{34} + w_{24},\ c_{23} + c_{44} + w_{24} \}$

$\Rightarrow k = 3, 4$ $\quad r_{ij} = 3$

$= \min \{ 0 + 3 + 5,\ 3 + 0 + 5 \}$ $\quad r_{24} = 3$

$= \min \{ 8, 8 \}$

$= 8$ (let us take $k = 3$)

$w_{03} = w_{02} + p_3 + q_3$

$= 12 + 1 + 1$

$= 14$

$c_{03} = \min_{0<k\le 3} \{ c_{ik-1} + c_{kj} + w_{ij} \}$

$\Rightarrow k = 1, 2, 3$ $\quad r_{03} = 2$

$= \min \{ c_{00} + c_{13} + w_{13},\ c_{01} + c_{23} + w_{13},\ c_{02} + c_{33} + w_{13} \}$

$= \min \{ 0 + 12 + 9,\ 8 + 3 + 9,\ 19 + 3 + 9 \}$

$= \min \{ 21, 20, 31 \}$

$= 20$ (for $k = 2$)

$w_{14} = w_{13} + p_4 + q_4$

$= 9 + 1 + 1$

$= 11$

$c_{14} = \min_{1<k\le 4} \{ c_{11} + c_{24} + w_{14},\ c_{12} + c_{34} + w_{14},\ c_{13} + c_{44} + w_{14} \}$

$\Rightarrow k = 2, 3, 4$ $\quad r_{14} = 2$

$= \min \{ 0 + 8 + 11,\ 7 + 3 + 11,\ 12 + 3 + 11 \}$

$= \min \{ 19, 21, 26 \}$

$= 21$ (for $k = 2$)

$w_{ij} = w_{i,j-1} + p_j + q_j$

$w_{04} = w_{03} + p_4 + q_4$

$= 14 + 1 + 1$

$= 16.$

$c_{ij} = \min_{i<k\le j} \{ c_{i,k-1} + c_{kj} + w_{ij} \}$

$c_{04} = \min_{0<k\le 4} \{ c_{i,k-1} + c_{kj} + w_{ij} \}$

$\Rightarrow k = 1, 2, 3, 4$

$c_{04} = \min \{ c_{00} + c_{14} + w_{04},\ c_{01} + c_{24} + w_{04},\ c_{02} + c_{34} + w_{04},\ c_{03} + c_{44} + w_{04} \}$

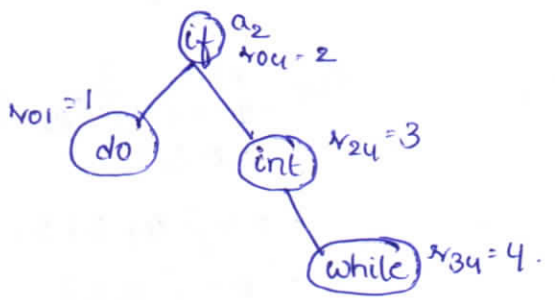
$= \min \{ 0 + 19 + 16,\ 8 + 5 + 16,\ 19 + 3 + 16,\ 25 + 0 + 16 \}$

$= \min \{ 35, 29, 38, 41 \}$

$= 32$ (for $k = 2$)

$\therefore r_{04} = 2.$

- The tree T_{04} has a root r_{04} is 2. Hence a_2 becomes root node



Algorithm:

```
Algorithm OBST(P, q, n)
{
  for i := 1 to n+1 do
  {
    w[i,i] := q_i;
    c[i,i] := 0;
  }
  for [m := 1 to n do
  {
   for i := 1 to n-m+1 do
   {
     j = i+m-1;
     c[i,j] := ∞
     w[i,j] := w[i,j-1] + p_j + q_j;
    for [r := i to j do
    {
      k = c[i,r-1] + c[r+1,j] + w[i,j];
     if (k < c[i,j]) then
     {
       c[i,j] := k;
       root[i,j] := r;
}}}}}
```

- The time complexity of this algorithm is $O(n^3)$.

0/1 Knapsack Problem:

If we are given n objects & a knapsack or a bag in which the object i that has weight w_i is to be placed. The knapsack has a capacity w. Then the profit can be earned is $P_i x_i$. The objective is to obtain filling of knapsack with maximum profit earned.

To solve this problem using Dynamic programming method. we will perform following steps:

1. Merging: we use merging operation to obtain the set S^{i+1} from S_i & S_i^1 as follows:

Initially $S^0 = \{0,0\}$ $\boxed{S^{i+1} = S^i \cup S_1^i}$

2. Purging:
   (i) If S^{i+1} contains (P_j, w_j) and (P_k, w_k) these two pairs such that $P_j \leq P_k$ & $w_j \geq w_k$ then (P_j, w_j) can be eliminated.
   (ii) In puging rule basically the dominated tuples gets purged.
   (iii) In short, remove the pair with less profit & more weight

Eg-1: Consider the knapsack for the instance n = 4.

$(w_1, w_2, w_3, w_4) = (10, 15, 6, 9)$ & $(P_1, P_2, P_3, P_4) = (2, 5, 8, 1)$ & $m = 21$.

Sol:

P_i	w_i
2	10
5	15
8	6
1	9

$S^i = \{P_i, w_i\}$

Initially

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

$S_1^0 = \{$ Select next pair of (P,w) & add it with $S^0 \}$

$= \{(2,10)\}$

$S^1 = \{$ Merge S^0 & $S_1^0 \}$ $\quad [S^{i+1} = S^i \cup S_1^i]$

$= \{(0,0)\ (2,10)\}$

$S_1^1 = \{$ Select next Pair of (P,w) & add it with $S^1 \}$

$= \{(5,15)\ (7,25)\}$

$S^2 = \{ (0,0) (2,10) (5,15) (7,25) \}$

$S_1^2 = \{$ Add next pair of (P,W) & add it with $S^2 \}$

$= \{(8,6)(10,16)(13,21)(15,31)\}$

$S^3 = \{$ Merge S^2 & $S_1^2 \}$

$= \{(0,0)\ (2,10)\ (5,15)\ (8,6)\ (10,16)\ (13,21)\ (15,31)\}$

Pair (7,25) is eliminated from S^3 because of purging rule let us assume $(P_j, w_j) = (7,25)$ and $(P_k, w_k) = (8,6)$. Hence $P_j \leq P_k$ & $w_j > w_k$ is true hence we will eliminate pair (P_j, w_j) ie, (7,25)

Now $S_1^3 = \{$ Add next pair of (P,W) & add it with $S^3 \}$

$= \{(1,9)\ (3,19)\ (6,24)\ (9,15)\ (11,25)\ (14,30)\ (16,40)\}$

$S^4 = \{$ Merge S^3 & $S_1^3 \}$

$= \{ (0,0)(2,10)(5,15)(8,6)(10,16)(13,21)(15,31)(1,9)(3,19)$
$(6,24)(9,15)(11,25)(14,30)(16,40)\}$

Now, S^4 contains all possible solutions of (P,W) pairs.

We will consider the knapsack capacity $W = 21$.

So, we will select pair (13, 21)

$(13,21) \in S^3$.

Hence set 3 is (8,6)

$\therefore$ Remaining weight $= (13-8, 21-6)$

$= (5,15) \in S^2$.

Hence set 2 is (5,15)

$\therefore$ Remaining weight $= (5-5, 15-15)$

$= (0,0)$

So no weight remaining

So, we get final solution as $\{0, 1, 1, 0\}$

Hence maximum profit $= 5+8 = 13$

Algorithm:

```
Algorithm DKP(P, w, n, m)
{
  S^0 := {0,0};
  for i := 1 to n-1 do
  {
    S_1^{i-1} = {(P,w) | (P-P_i, w-w_i) ∈ S^{i-1} and w ≤ m};
    S^i := MergePurge(S^{i-1}, S_1^{i-1});
  }
  (P_x, w_x) = last pair in S^{n-1};
  (P_y, w_y) = (P' + P_n, w' + w_n);
  if (P_x > P_y) then
    x_n := 0;
  else
    x_n := 1;
  TraceBackFor(x_{n-1} -- x_1);
}
```

All pairs Shortest path Problem:

Suppose if given graph is weighted graph. Then objective is to find shortest path between each & every pair of nodes present in given graph.

Procedure:

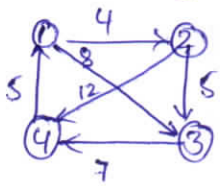
* To find shortest path between every pair of nodes using dynamic Programming,

1. we need to find A_{ij}^{K} for K = 0 to n. A_{ij}^{K} represents path between i to j via K.

2. $A^0 = w[i,j]$

$$A_{ij}^{K} = \min\{A_{ij}^{K-1}, A_{ik}^{K-1} + A_{kj}^{K-1}\}$$

Eg-1:



Now

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

Now to calculate A^1, do the following

$A^1_{23} = \min\{A^0_{23}, A^0_{21} + A^0_{13}\} = \min\{5, \infty + 8\} = 5$

$A^1_{32} = \min\{A^0_{32}, A^0_{31} + A^0_{12}\} = \min\{\infty, \infty + 4\} = \infty.$

$A^1_{24} = \min\{A^0_{24}, A^0_{21} + A^0_{14}\} = \min\{12, \infty + \infty\} = 12$

$A^1_{42} = \min\{A^0_{42}, A^0_{41} + A^0_{12}\} = \min\{\infty, 5+4\} = 9.$

$A^1_{34} = \min\{A^0_{34}, A^0_{31} + A^0_{14}\} = \min\{7, \infty + \infty\} = 7.$

$A^1_{43} = \min\{A^0_{43}, A^0_{41} + A^0_{13}\} = \min\{\infty, 5+8\} = 13.$

Now A^1 becomes

$$\begin{bmatrix} 0 & 4 & 8 & \infty \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & 9 & 13 & 0 \end{bmatrix}$$

To get A^2, calculate following.

$A^2_{13} = \min\{A^1_{13}, A^1_{12} + A^1_{23}\} = \min\{8, 4+5\} = 8$

$A^2_{14} = \min\{A^1_{14}, A^1_{12} + A^1_{24}\} = \min\{\infty, 4+12\} = 16.$

$A^2_{31} = \min\{A^1_{31}, A^1_{32} + A^1_{21}\} = \min\{\infty, \infty + \infty\} = \infty.$

$A^2_{34} = \min\{A^1_{34}, A^1_{32} + A^1_{24}\} = \min\{7, \infty + 12\} = 7.$

$A^2_{41} = \min\{A^1_{41}, A^1_{42} + A^1_{21}\} = \min\{5, \infty + \infty\} = 5$

$A^2_{43} = \min\{A^1_{43}, A^1_{42} + A^1_{23}\} = \min\{\infty, \infty + 5\} = \infty.$

A^2 becomes

$$\begin{bmatrix} 0 & 4 & 8 & 16 \\ \infty & 0 & 5 & 12 \\ \infty & \infty & 0 & 7 \\ 5 & 9 & 13 & 0 \end{bmatrix}$$

To get A^3,

$A^3_{12} = \min\{A^2_{12}, A^2_{13} + A^2_{32}\} = \min\{4, 8+\infty\} = 4.$

$A^3_{21} = \min\{A^2_{21}, A^2_{23} + A^2_{31}\} = \min\{\infty, 5+\infty\} = \infty.$

$A^3_{14} = \min\{A^2_{14}, A^2_{13} + A^2_{34}\} = \min\{16, 8+7\} = 15$

$A^3_{41} = \min\{A^2_{41}, A^2_{43} + A^2_{31}\} = \min\{5, 13+\infty\} = 5$

$A^3_{24} = \min\{A^2_{24}, A^2_{23} + A^2_{34}\} = \min\{12, 5+7\} = 12$

$A^3_{42} = \min\{A^2_{42}, A^2_{43} + A^2_{32}\} = \min\{9, 13+\infty\} = 9.$

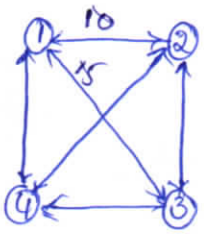
The Travelling Salesperson problem:

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

From the principal of optimality it follows that,

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

Eg:



The edge lengths are given by the matrix,

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

Sol: First we select any arbitary vertex. Say vertex 1.

So destination vertex is 1.

Step-1: Consider $S = \phi$.

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

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

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

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

Step-2: Consider $S = 1$.

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

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

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

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

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

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

Now A^3 becomes,

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

To get A^4,

$A^4_{12} = \min\{ A^3_{12}, A^3_{14} + A^3_{42} \} = \min\{4, 16+9\} = 4$

$A^4_{21} = \min\{ A^3_{21}, A^3_{24} + A^3_{41} \} = \min\{\infty, 12+5\} = 17$

$A^4_{23} = \min\{ A^3_{23}, A^3_{24} + A^3_{43} \} = \min\{5, 12+13\} = 5$

$A^4_{32} = \min\{ A^3_{32}, A^3_{34} + A^3_{42} \} = \min\{\infty, 7+9\} = 16$

$A^4_{13} = \min\{ A^3_{13}, A^3_{14} + A^3_{43} \} = \min\{8, 16+13\} = 8$

$A^4_{31} = \min\{ A^3_{31}, A^3_{34} + A^3_{41} \} = \min\{\infty, 7+5\} = 12$

Now A^4 becomes,

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

Now A^4 gives shortest distance b/w any pair of vertices.

Algorithm:

```
Algorithm Allpaths(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]);
}
```

Complexity:

The time complexity for this algorithm is $O(V^3)$

(V is the number of vertices in the graph)

Step-3: we compute $g(i,S)$ with $|S| = 2$.

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

$$g(3,\{2,4\}) = \min\{ c_{23} + g(2,\{4\}), c_{34} + g(4,\{2\}) \}$$
$$= \min\{ 13+18, 12+13 \}$$
$$= 25$$

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

Step-4: Consider $S = 3$.

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

Thus the optimal tour is of path length 35.

Consider step-4, now from vertex 1 we obtain the optimum path as $d(1,2)$. Hence select vertex 2. Now consider step 3, in which from vertex 2, we obtain optimum cost from $d(2,4)$. Hence select vertex 4. Now in step 2 we get remaining vertex 3 as $d(4,3)$ is optimum.

Hence optimal tour is 1, 2, 4, 3, 1

Algorithm

```
Algorithm TSP(V, E)
{
  C({1}, 1) := 0;
  for S := 2 to n do
    for all subsets S ⊆ {1,2..n} of size S and containing 1;
      C(S,1) := ∞;
    for all j ∈ S, j ≠ 1:
      g(j, S) = min{ c_ij + g(j, S-{j}) };
  } return min(g(i,S));
```

— The total running time is $O(n^2 \cdot 2^n)$.

Reliability Design: Consider a system of n devices $d_1, d_2 \ldots d_n$ connected serially

The objective of this concept is,

$$\text{maximize} \prod_{1 \le i \le n} \phi_i m_i \text{ subject to } \sum_{1 \le i \le n} c_i m_i \le C$$

$\prod_{1 \le i \le n} \phi_i m_i$ represents total reliability of the system.

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

Here
r_i - Reliability of individual device
m_i - No. of devices of same type D_i.

$\sum_{1 \le i \le n} c_i m_i$ represents total cost of the n states system.

$$u_i = c + c_i - \sum_{j=1}^{n} c_j / c_i$$

u_i represents maximum no of same devices
c_i " individual cost of device D_i.

Eg: Consider 3 stage system of devices D_1, D_2, D_3 of costs Rs 30, 15, 20 respectively on reliabilities 0.9, 0.8, 0.5 & cost of system is 105/-

Sol: Consider Devices (D_1, D_2, D_3)

$(c_1, c_2, c_3) = (30, 15, 20)$

$c = 105$.

$(r_1, r_2, r_3) = (0.9, 0.8, 0.5)$

$u_i = c + c_i - \sum_{j=1}^{n} c_j / c_i$

$u_1 = 105 + 30 - (30+15+20)/30 = 2.33 \cong 2$.

$u_2 = 105 + 15 - (30+15+20)/15 \cong 3$.

$u_3 = 105 + 20 - (30+15+20)/20 = 3$.

$(u_1, u_2, u_3) = (2, 3, 3)$

Initially $S^0 = \{(1,0)\}$ Here $i = 1$, $m_i = 1$

$\phi_1 m_1 = 1 - (1 - 0.9)^1 = 0.9$.

S_1^1 calculated as $S^0 = \{(1,0)\}$

$r = (0.9 \times 1)$ and $c_1 = 30 + 0 = 30$.

$S_1 = ((0.9 \times 1), (30+0))$

$S_1' = \{(0.9, 30)\}$

$S_2' = \{(0.99 \times 1)\ (60+0)\}$

$= \{(0.99, 60)\}$

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

$= 1 - (1 - 0.9)^2 = 0.99$

$c_i m_i = 2 \times 30 = 60$

$S' =$ Merge S_1' & S_2'

$= \{(0.9, 30)(0.99, 60)\}$

Now,

$u_2 = 8$

$S_1^2 = \{(0.9 \times 0.8, 30+15),$
$(0.99 \times 0.8, 60+15)\}$

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

$= 1 - (1 - 0.8)^1 = 0.8$

$c_i m_i = 1 \times 15 = 15$

$S_1^2 = \{(0.72, 45), (0.792, 75)\}$

$S_2^2 = \{(0.9 \times 0.96, 30+30)(0.99 \times 0.96, 60+30)\}$

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

$\phi_i m_i = 1 - (1 - r_i)^m$

$= 1 - (1 - 0.8)^2 = 0.96$

$c_i m_i = 2 \times 15 = 30$

$S_3^2 = \{(0.9 \times 0.992, 30+45)(0.99 \times 0.992, 60+40)\}$

$= \{(0.8928, 75)(0.98208, 105)\}$

$\phi_i m_i = 1 - (1 - r_i)^m$

$= 1 - (1 - 0.8)^3 = 0.992$

$c_i m_i = 15 \times 3 = 45$

$S^2 =$ Merge S_1^2, S_2^2 & S_3^2

$= \{(0.792, 45)(0.792, 75)(0.864, 60)(0.9504, 90)$
$(0.8928, 75)(0.98208, 105)\}$

In S^2, with same cost of 75, we have 2 different realibilities. So, eliminate the pair with less reliability using domenance rule & also eliminate (0.9504, 90)

Hence

$S^2 = \{(0.72, 45)(0.792, 75)(0.864, 60)(0.8928, 75)(0.98208, 105)\}$

Now,

$u_3 = 3$

$S_1^3 = \{(0.72 \times 0.5, 45+20)$
$(0.864 \times 0.5, 60+20)$
$(0.8928 \times 0.5, 75+20)$
$(0.98208 \times 0.5, 105+20)\}$

$\phi_i m_i = 1-(1-r_i)^{m_i}$
$= 1-(1-0.5)^1 = 0.5$
$c_i m_i = 15 \times 1 = 15$

$= \{(0.36, 65)(0.432, 80)(0.4464, 95)\}$ [last pair is eliminated since the cost is exceeded by 20]

$S_2^3 = \{(0.72 \times 0.75, 45+40)$
$(0.864 \times 0.75, 60+40)$
$(0.8928 \times 0.5, 75+40)$
$(0.98208 \times 0.5, 105+40)\}$

$\phi_i m_i = 1-(1-r_i)^{m_i}$
$= 1-(1-0.5)^2 = 0.75$
$c_i m_i = 15 \times 2 = 30$

$= \{(0.54, 85)(0.648, 100)\}$ [last two pairs are eliminated since the total cost is exceeded]

$S_3^3 = \{(0.72 \times 0.875, 45+60)\}$
$= \{(0.63, 105)\}$

$\phi_i m_i = 1-(1-r_i)^{m_i}$
$= 1-(1-0.5)^3 = 0.875$
$c_i m_i = 3 \times 15 = 45$.

S^3 = Merge S_1^3, S_2^3 & S_3^3.

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

(0.4464, 95) & (0.63, 105) pairs are eliminated using dominance rule [(f_1, x_1) dominates (f_2, x_2) if $f_1 \geq f_2$ for $x_1 \leq x_2$ then eliminate (f_2, x_2)]

Finally

$S^3 = \{(0.36, 65)(0.432, 80)(0.54, 85)(0.648, 100)\}$

Given total cost = 105

So consider pair (0.648, 100) which is present in S_2^3. So $m_3 = 2$

The pair (0.648, 100) obtained from (0.864, 60) present in S_2^2. So $m_2 = 2$

The pair (0.864, 60) obtained from (0.9, 30) present in S_1^1. So $m_1 = 1$

So, finally

$m_1 = 1$
$m_2 = 2$
$m_3 = 2$