DYNAMIC PROGRAMMING

Dynamic programming was invented by the U.S. Mathematician Richard Bellman in 1950.

It is applied to optimization problems.

This is an algorithm design method that can be used when the solution to a problem viewed as a sequence of decisions.

It obtains the solution using "principle of optimality"

It states that "In an optimal sequence of decisions or choices, each subsequence must also be optimal".

Applications

1. 0/1 Knapsack
2. Matrix chain multiplication.
3. Multistage Graphs.
4. Optimal Binary search trees
5. All pairs shortest path problem
6. Travelling sales person problem.
7. Reliability design.

Multistage Graphs

A multistage graph $G = (V, E)$ is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets V_i, $1 \leq i \leq k$.

If (u, v) is an edge in E then $u \in V_i$ and $v \in V_{i+1}$ for some i, $1 \leq i < k$.

The sets V_1 and V_k are such that $|V_1| = |V_k| = 1$

Let s and t be the vertices in V_1 and V_k.

The vertex s is the source and t is the sink.

Let $c(i, j)$ be the cost of edge (i, j).

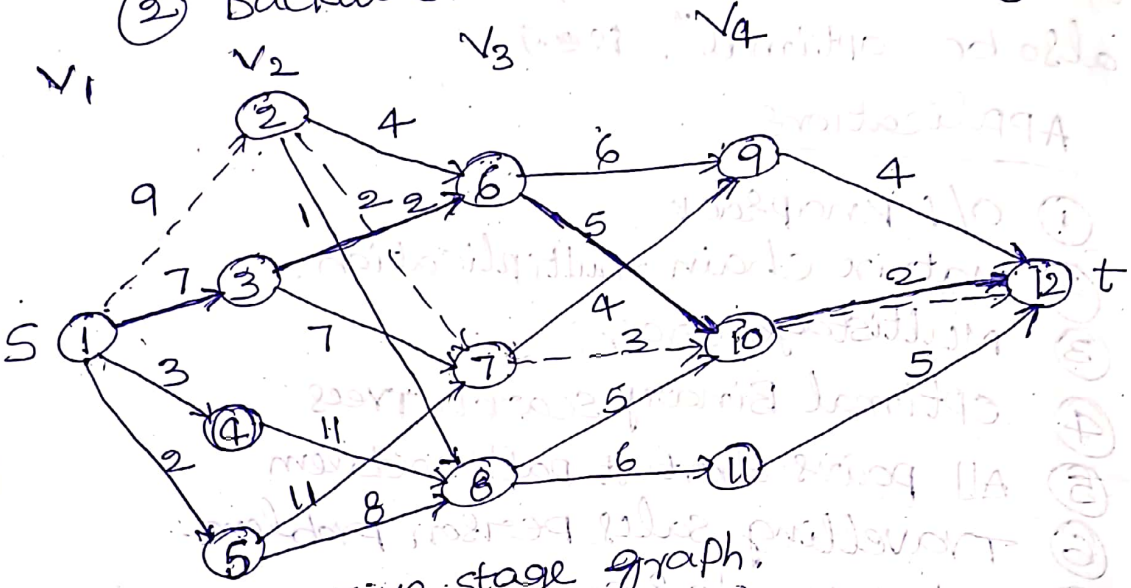
The cost of a path from s to t is the sum of the costs of edges on the path.

In this multistage graph, we need to find a minimum-cost path from S to t.

The vertex set V_i defines a stage in the graph.

We use two approaches to find the minimum cost path from source S to the sink t: They are.

1. Forward approach
2. Backward approach.



Five stage graph.

(1) Forward approach :-

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

$D(i,j) = l$ Here l is the node that minimises the $c(j,l) + cost(i+1,l)$

i is the stage number and j is the vertex number.

$c(j,l)$ minimum edge cost from vertex j to vertex l in the next stage.

$cost(i+1,l)$ is the cost of vertex l in the $i+1$th stage. (present stage is i)

In the forward approach we start at sink vertex.

$Cost(5,12) = 0$

Step 1:

$Cost(4,9) = \min\{C(9,12) + Cost(5,12)\}$
$= \min\{4+0\} = 4$

$Cost(4,10) = \min\{C(10,12) + Cost(5,12)\}$
$= \min\{2+0\} = 2.$

$Cost(4,11) = \min\{C(11,12) + Cost(5,12)\}$
$= \min\{5+0\} = 5.$

Step 2:

$Cost(3,6) = \min\{C(6,9) + Cost(4,9),$
$C(6,10) + Cost(4,10)\}$
$= \min\{6+4, 5+2\} = 7$
$D(3,6) = 10$

$Cost(3,7) = \min\{C(7,9) + Cost(4,9),$
$C(7,10) + Cost(4,10)\}$
$= \min\{4+4, 3+2\} = 5$
$D(3,7) = 10$

$Cost(3,8) = \min\{C(8,10) + Cost(4,10)$
$C(8,11) + Cost(4,11)\}$
$= \min\{5+2, 6+5\} = 7$
$D(3,8) = 10.$

Step 3:

$Cost(2,2) = \min\{C(2,6) + Cost(3,6),$
$C(2,7) + Cost(3,7),$
$C(2,8) + Cost(3,8)\}$
$= \min\{4+7, 2+5, 1+7\} = 7$
$D(2,2) = 7$

$cost(2,3) = \min\{ c(3,6) + cost(3,6),\ c(3,7) + cost(3,7) \}$

$= \min\{2+7, 7+5\} = 9$

$D(2,3) = 6$

$cost(2,4) = \min\{ c(4,8) + cost(3,8) \}$

$= \min\{11+7\} = 18$

$D(2,4) = 8$

$cost(2,5) = \min\{ c(5,7) + cost(3,7),\ c(5,8) + cost(3,8) \}$

$= \min\{11+5, 8+7\} = 15$

$D(2,5) = 8$

Step 4:

$cost(1,1) = \min\{ c(1,2) + cost(2,2),\ c(1,3) + cost(2,3),\ c(1,4) + cost(2,4),\ c(1,5) + cost(2,5) \}$

$= \min\{9+7, 7+9, 3+18, 2+15\}$

$= 16$

$D(1,1) = 2$ or 3

$D(1,1) = 2$
$D(2,2) = 7$
$D(3,7) = 10$
$D(4,10) = 12$

this is $1-2-7-10-12$
Total cost = 16

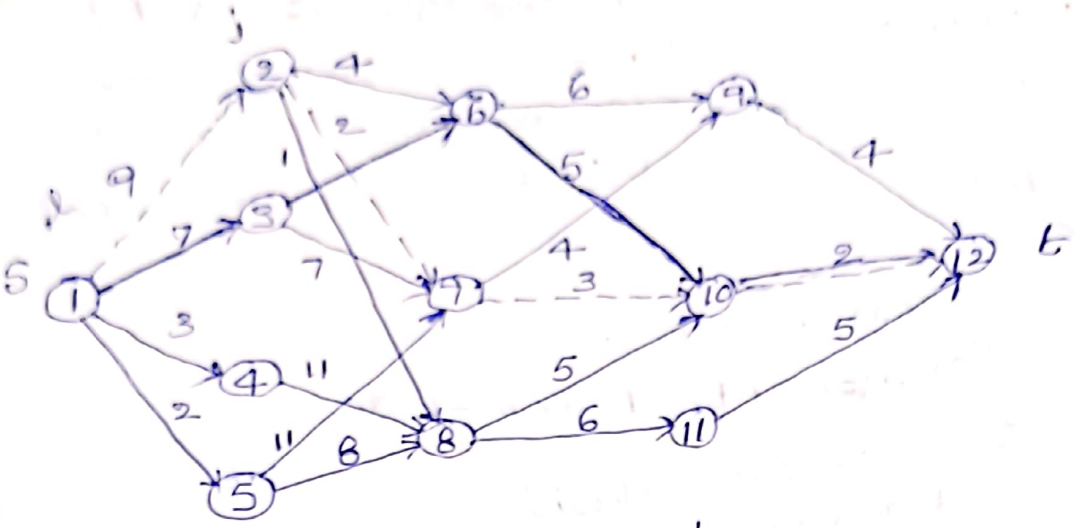
$D(1,1) = 3$
$D(2,3) = 6$
$D(3,6) = 10$
$D(4,10) = 12$

path is
$1-3-6-10-12$
total cost = 16

```
Algorithm FGraph(G, k, n, P)
// The input is a k-stage graph G = (V, E)
// with n vertices 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 a vertex such that <j,r> is an
    edge of G and c[j,r]+cost[r] is minimu
    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]];
}
```

Multistage Graph Pseudocode for forward approach.

Time complexity = $O(n^2)$.



Five stage graph.

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

Let bP(i,j) be a minimum cost path from vertex s to a vertex j in V_i.

Let bcost(i,j) be the cost of bP(i,j).

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

$D(i,j) = l$. Here l is the node that minimizes $bcost(i-1,l) + c(l,j)$

Backward approach starts with source vertex.

$$bcost(1,1) = 0$$

Step 1:

$$bcost(2,2) = \min\{ bcost(1,1) + c(1,2) \}$$
$$= \min\{ 0+9 \} = 9$$

$$bcost(2,3) = \min\{ bcost(1,1) + c(1,3) \}$$
$$= \min\{ 0+7 \} = 7$$

$$bcost(2,4) = \min\{ bcost(1,1) + c(1,4) \}$$
$$= \min\{ 0+3 \} = 3$$

$$bcost(2,5) = \min\{ bcost(1,1) + c(1,5) \}$$
$$= \min\{ 0+2 \}.$$

Step 2:

$bcost(3,6) = \min\{bcost(2,2)+c(2,6),\ bcost(2,3)+c(3,6)\}$

$= \min\{9+4, 7+2\} = 9$

$D(3,6) = 3$

$bcost(3,7) = \min\{bcost(2,2)+c(2,7),\ bcost(2,3)+c(3,7),\ bcost(2,5)+c(5,7)\}$

$= \min\{9+2, 7+7, 2+11\} = 11$

$D(3,7) = 2.$

$bcost(3,8) = \min\{bcost(2,2)+c(2,8),\ bcost(2,4)+c(4,8),\ bcost(2,5)+c(5,8)\}$

$= \min\{9+1, 3+11, 2+8\} = 10$

$D(3,8) = 2$ or 5

Step 3:

$bcost(4,9) = \min\{bcost(3,6)+c(6,9),\ bcost(3,7)+c(7,9)\}$

$= \min\{9+6, 11+4\} = 15$

$D(4,9) = 6$ or 7

$bcost(4,10) = \min\{bcost(3,6)+c(6,10),\ bcost(3,7)+c(7,10),\ bcost(3,8)+c(8,10)\}$

$= \min\{9+5, 11+7, 10+5\} = 14$

$D(4,10) = 6$ or $7.$

$bcost(4,11) = \min\{bcost(3,8)+c(8,11)\} =$

$\min\{10+6\} = 16.$

$D(4,11) = 8.$

Step 4:

$bcost(5,12) = \min\{ bcost(4,9)+C(9,12),$
$bcost(4,10)+C(10,12),$
$bcost(4,11)+C(11,12)\}$
$= \min\{ 15+4, 14+2, 16+5\} = 16$

$D(5,12) = 10$.

$D(5,12) = 10$
$D(4,10) = 6$
$D(3,6) = 3$
$D(2,3) = 1$
So Path is
1-3-6-10-12
Cost = 7+2+5+2 = 16

$D(5,12) = 10$
$D(4,10) = 7$
$D(3,7) = 2$
$D(2,2) = 1$
So path is
1-2-7-10-12
Cost = 9+2+3+2 = 16.

```
Algorithm BGraph(G, k, n, P)
// Same function as FGraph
{
    bcost[1] := 0.0;
    for j := 2 to n do
    {
        // Compute bcost[j].
        Let r be such that <r, j> is an edge
        of G and 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]];
}
```

Multistage graph pseudocode for Backward Approach =

All pairs shortest path (Floyd warshall Algorithm)

For a given graph G = (V, E), we have to find out shortest path between all pairs of nodes.

We need find out shortest path from each node to every other node in a given graph.

To find out the shortest path, we use following formula.

$$A^k[i,j] = \min\{A^{k-1}[i,j], A^{k-1}[i,k] + A^{k-1}[k,j]\}$$

k is the number of intermediate vertices.

Here we need to chose either the direct path from i to j or i to j via k based on the minimum value.

Find the shortest path between all pairs of nodes in a graph

$$A^0 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{bmatrix} \quad A^1 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

$$A^3 \begin{bmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

Stage 1 $\quad K = 1$

$A^1[1,1] = 0$

$A^1[1,2] = \min\{A^0[1,2], A^0[1,1] + A^0[1,2]\}$

$A^1[1,3] = \min\{8, 0+8\} = 8$

$A^1[2,1] = \min\{A^0[1,3], A^0[1,1] + A^0[1,3]\}$
$= \min\{5, 0+5\} = 5$

$A^1[2,2] = 0$

$A^1[2,3] = \min\{A^0[2,1], A^0[2,1] + A^0[1,1]\}$
$= \min\{3, 3+0\} = 3$

$A^1[3,1] = \min\{A^0[2,3], A^0[2,1] + A^0[1,3]\} = \min\{\infty, 3+5\}$
$= 8.$

$A^1[3,2] = \min\{A^0[3,1], A^0[3,1] + A^0[1,1]\} = \min\{\infty, \infty+0\}$
$= \infty.$

$A^1[3,3] = 0$

$A^1[3,2] = \min\{A^0[3,2], A^0[3,1] + A^0[1,2]\} = \min\{2, \infty+$

stage = 2, K = 2.

$A^2[1,2] = \min\{A^1[1,2], A^1[1,2]+A^1[2,2]\}$
$= \min\{8, 8+0\} = 8.$

$A^2[1,3] = \min\{A^1[1,3], A^1[1,2]+A^1[2,3]\}$
$= \min\{5, 8+8\} = 5$

$A^2[2,1] = \min\{A^1[2,1], A^1[2,2]+A^1[2,1]\}$
$= \min\{3, 0+3\} = 3$

$A^2[2,3] = \min\{A^1[2,3], A^1[2,2]+A^1[2,3]\}$
$= \min\{8, 0+8\} = 8.$

$A^2[3,1] = \min\{A^1[3,1], A^1[3,2]+A^1[2,1]\}$
$= \min\{5, 2+3\} = 5$

$A^2[3,2] = \min\{A^1[3,2], A^1[3,2]+A^1[2,2]\}$
$= \min\{2, 2+0\} = 2.$

$$A^2 = \begin{bmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

stage = 3, K = 3

$A^3[1,2] = \min\{A^2[1,2], A^2[1,3]+A^2[3,2]\}$
$= \min\{8, 5+2\} = 7$

$A^3[1,3] = \min\{A^2[1,3], A^2[1,3]+A^2[3,3]\}$
$= \min\{5, 5+0\} = 5$

$A^3[2,1] = \min\{A^2[2,1], A^2[2,3]+A^2[3,1]\}$
$= \min\{3, 8+5\} = 3$

$A^3[2,3] = \min\{A^2[2,3], A^2[2,3]+A^2[3,3]\}$
$= \min\{8, 8+0\} = 8.$

$A^3[3,1] = \min\{A^2[3,1], A^2[3,3]+A^2[3,1]\}$
$= \min\{5, 0+5\} = 5$

$A^3[3,2] = \min\{A^2[3,2], A^2[3,3]+A^2[3,2]\}$
$= \min\{2, 0+2\} = 2$

the answered nodes in a

$$A^3 = \begin{bmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{bmatrix}$$

This A^3 matrix contains the all pairs shortest paths of the given graph.
Time complexity is $O(n^3)$.

Single source shortest path [Bellman-Ford Algorithm].

// Algorithm Allpaths(cost, A, n)
// cost[1:n, 1:n] is the cost adjacency matrix
// of a graph with n vertices; A[i,j] is the cost
// of a shortest path from vertex i to vertex j.
// cost[i,i] = 0.0 for 1≤i≤n.

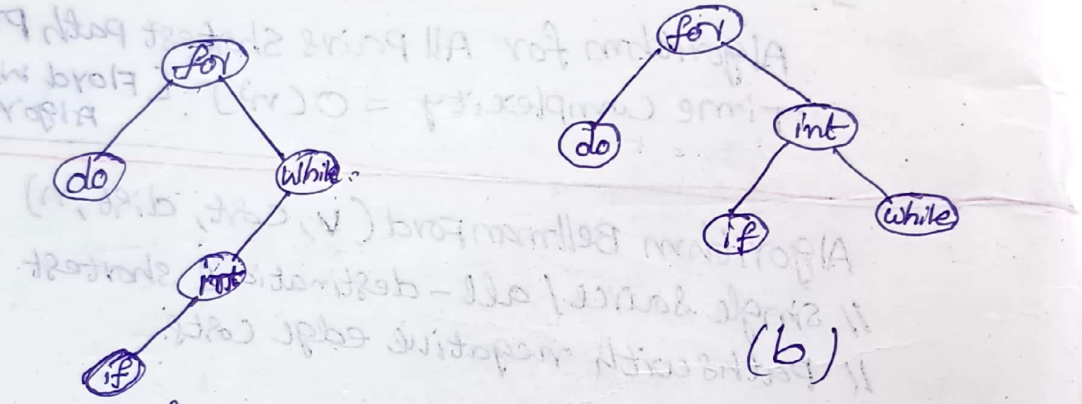
```
{
  for i:=1 to n do
    for j:=1 to n do
      A[i,j]:=cost[i,j]; // copy cost into A.

  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]]

}
```

Algorithm for All pairs shortest path problem.
Time complexity = $O(n^3)$. [Floyd Warshall Algorithm]

Optimal Binary Search Trees (OBST)

Given a fixed set of identifiers, we have to create a binary search tree with less number of comparisions. The OBST must be created so that it minimizes the searching for identifier (element).

Suppose, we have 5 identifiers such as for, do, while, int and if. We create two binary search trees as follows.



In figure a, we require four comparisions to search an identifier.

In figure b, we require only three comparisions to find an identifier.

To find an element, we have to search it from root node to leaf level.

The number of comparisions are always based on the number of levels that are present in the tree.

Different identifiers are searched with various probabilities

There is unsuccessful search in addition to successful search.

We have a set of identifiers $\{a_1, a_2 \ldots a_n\}$ with $a_1 < a_2 < \ldots < a_n$.

Let $P(i)$ be the probability of searching an element a_i.

Let $q(i)$ be the probability of searching the identifier x such that $a_i < x < a_{i+1}$, $0 \le i \le n$.

Then $\sum_{0 \le i \le n} q(i)$ is the probability of an unsuccessful search.

$$\sum_{1 \le i \le n} P(i) + \sum_{0 \le i \le n} q(i) = 1$$

With this data, we need to construct optimal binary search tree for set of identifiers $\{a_1, a_2, \ldots a_n\}$.

For a given n number of vertices, we have to construct tree T_{0n}.

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

$$w(i,j) = w(i, j-1) + P(j) + q(j)$$

$r(i,j)$ = k value for which $C[i,j]$ is minimum.

Ex:- n=4 (a_1, a_2, a_3, a_4) = (do, if, int, while)

$P(1:4) = (3, 3, 1, 1)$ $q(0:4) = (2, 3, 1, 1, 1)$

Set j - i = 0. For this case we calculate

$T_{00}, T_{11}, T_{22}, T_{33}$ and T_{44}.

T_{00}: $C(0,0) = 0$

$w(0,0) = q(0) = 2$

$r(0,0) = 0$

$C(i,i) = 0$
$r(i,i) = 0$
$w(i,i) = q(i)$

T_{00}	T_{11}	T_{22}	T_{33}	T_{44}
$C(0,0) = 0$	$C(1,1) = 0$	$C(2,2) = 0$	$C(3,3) = 0$	$C(4,4) = 0$
$w(0,0) = 2$	$w(1,1) = 3$	$w(2,2) = 1$	$w(3,3) = 1$	$w(4,4) = 1$
$r(0,0) = 0$	$r(1,1) = 0$	$r(2,2) = 0$	$r(3,3) = 0$	$r(4,4) = 0$

T_{01}	T_{12}	T_{23}	T_{34}
$C(0,1) = 8$	$C(1,2) = 7$	$C(2,3) = 3$	$C(3,4) = 3$
$w(0,1) = 8$	$w(1,2) = 7$	$w(2,3) = 3$	$w(3,4) = 3$
$r(0,1) = 1$	$r(1,2) = 2$	$r(2,3) = 3$	$r(3,4) = 4$

T_{02}	T_{13}	T_{24}
$C(0,2) = 19$	$C(1,3) = 12$	$C(2,4) = 8$
$w(0,2) = 19$	$w(1,3) = 9$	$w(2,4) = 5$
$r(0,2) = 1$	$r(1,3) = 2$	$r(2,4) = 3$

T_{03}	T_{14}
$C(0,3) = 25$	$C(1,4) = 19$
$w(0,3) = 14$	$w(1,4) = 11$
$r(0,3) = 2$	$r(1,4) = 2$

T_{04}
$C(0,4) = 32$
$w(0,4) = 16$
$r(0,4) = 2$

set $j-i=1$, we calculate

T_{01}, T_{12}, T_{23} and T_{34}

T_{01} $\quad w(0,1) = w(0,0) + P(1) + q(1)$

$= 2 + 3 + 3 = 8$

$c(0,1) = \min_{0<K\leq 1} \{c(0,0) + c(1,1)\} + w(0,1)$

$= \min_{0<K\leq 1} \{0+0\} + 8$

$K=1 \quad = 8.$

$r(0,1) = 1$

T_{12}: $\quad w(1,2) = w(1,1) + P(2) + q(2)$

$= 3 + 3 + 1 = 7$

$c(1,2) = \min_{1<K\leq 2} \{c(1,1) + c(2,2)\} + w(1,2)$

$K=2$

$= \min \{0+0\} + 7 = 7$

$r(1,2) = 2.$

T_{23}: $\quad w(2,3) = w(2,2) + P(3) + q(3)$

$= 1 + 1 + 1 = 3$

$c(2,3) = \min_{2<K\leq 3} \{c(2,2) + c(3,3)\} + w(2,3)$

$K=3$

$= \min \{0+0\} + 3 = 3$

$r(2,3) = 3$

T_{34}: $\quad w(3,4) = w(3,3) + P(4) + q(4)$

$= 1 + 1 + 1 = 3$

$c(3,4) = \min_{3<K\leq 4} \{c(3,3) + c(4,4)\} + w(3,4)$

$K=4$

$= \min \{0+0\} + 3 = 3$

$r(3,4) = 4$

set $j-i=2$, we calculate the values of T_{02}, T_{13}, T_{24}

T_{02}:

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

$$c(0,2) = \min_{0<k\leq 2} \left\{ \begin{matrix} c(0,0)+c(1,2) \\ c(0,1)+c(2,2) \end{matrix} \right\} + w(0,2)$$
$$= \min_{k=1,2} \left\{ \begin{matrix} 7 \\ 8 \end{matrix} \right\} + 12 = 19$$

$$r(0,2) = 1$$

T_{13}:

$$w(1,3) = w(4,2) + P(3) + q(3)$$
$$= 7 + 1 + 1 = 9$$

$$c(1,3) = \min_{1<k\leq 3} \left\{ \begin{matrix} c(1,1)+c(2,3) \\ c(1,2)+c(3,3) \end{matrix} \right\} + w(1,3)$$
$$= \min_{k=2,3} \left\{ \begin{matrix} 0+3 \\ 7+0 \end{matrix} \right\} + 9 = 12$$

$$r(1,3) = 2$$

T_{24}:

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

$$c(2,4) = \min_{2<k\leq 4} \left\{ \begin{matrix} c(2,2)+c(3,4) \\ c(2,3)+c(4,4) \end{matrix} \right\} + w(2,4)$$
$$= \min_{k=3,4} \left\{ \begin{matrix} 0+3 \\ 3+0 \end{matrix} \right\} + 5 = 8$$

$r(2,4) = 3$ or 4 But we select

$r(2,4) = 3$.

Next set $j-i=3$, we have

T_{03} and T_{14}.

T_{03}:

$$w(0,3) = w(0,2)+P(3)+q(3)$$
$$= 12+1+1 = 14.$$

$$C(0,3) = \min_{\substack{0<K\leq 3 \\ K=1,2,3}} \left\{ \begin{matrix} C(0,0)+C(1,3), \\ C(0,1)+C(2,3), \\ C(0,2)+C(3,3) \end{matrix} \right\} + w(0,3)$$

$$= \min \left\{ \begin{matrix} 0+12 \\ 8+3 \\ 19+0 \end{matrix} \right\} + 14 = 25$$

$$r(0,3) = 2$$

T_{14}: $w(1,4) = w(1,3)+P(4)+q(4)$

$$= 9+1+1 = 11$$

$$C(1,4) = \min_{\substack{1<K\leq 4 \\ K=2,3,4}} \left\{ \begin{matrix} C(1,1)+C(2,4) \\ C(1,2)+C(3,4) \\ C(1,3)+C(4,4) \end{matrix} \right\} + w(1,4)$$

$$= \min \left\{ \begin{matrix} 0+8 \\ 7+3 \\ 12+0 \end{matrix} \right\} + 11 = 19$$

$$r(1,4) = 2$$

Next set $j-i=4$, we have calculate

T_{04}

If there are n number of identifiers then we will construct T_{0n}.

Here we have 4 identifiers, so we construct up to T_{04}.

To4:

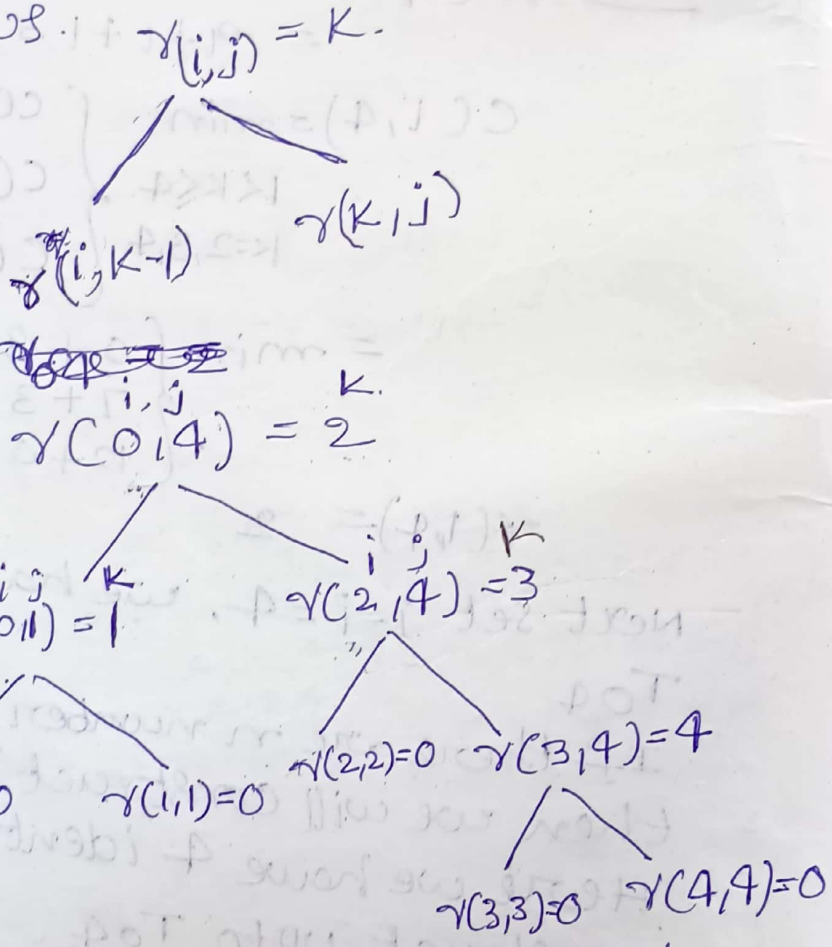
$$w(0,4) = w(0,3) + P(4) + Q(4)$$
$$= 19 + 1 + 1 = 16$$

$$C(0,4) = \min_{\substack{0<K\leq 4 \\ K=1,2,3,4}} \left\{ \begin{matrix} C(0,0)+C(1,4), \\ C(0,1)+C(2,4) \\ C(0,2)+C(3,4) \\ C(0,3)+C(4,4) \end{matrix} \right\} + w(0,4).$$

$$= \min \left\{ \begin{matrix} 0+19 \\ 8+8 \\ 19+4 \\ 25+0 \end{matrix} \right\} + 16 = 32$$

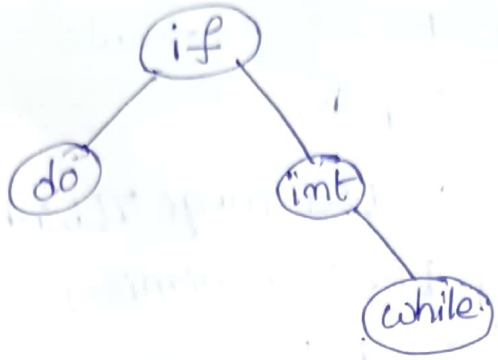
$r(0,4) = 2$

After computing all these values, we need to construct optimal binary search Tree as follows.



Expand the tree until all the leaf nodes will ~~become~~ have k value 0.

$(a_1, a_2, a_3, a_4) = (do, if, int, while)$
1 2 3 4



This is the final optimal binary search tree.

```
Algorithm OBST(P, q, n)
// Given n distinct identifiers a1<a2<...an and
// probabilities P[i], 1≤i≤n and
// q[i], 0≤i≤n. This algorithm computes
// the cost c[i,j] of optimal binary search
// trees tij for identifiers a(i+1), .... aj.
// It also computes r[i,j], the root of tij
// w[i,j] is the weight of tij.
{
  for(i:=0 to n-1 do
    { // Initialization
      w[i,i]:= q[i]; r[i,i]:=0; c[i,i]=0;
      // optimal trees with one node.
      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;
  for m:=2 to n do // Find optimal trees with m
  for i:=0 to n-m do                    nodes.
  {
    j:= i+m;
    w[i,j]:= w[
```

// calculate c[i,j] by using formula.

```
K := Find(C, r, i, j);
// A value of l in the range r[i,j-1] ≤ l
// ≤ r[i+1, j] that minimizes
// c[i,l-1] + c[l,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;
}
```

The time complexity is $O(n^3)$

We requires to compute c[i,j] for (j-i) = 1, 2, ... n in the order.
when j - i = m, there are n - m + 1 c[i,j]'s to compute. The computation of each of these c[i,j]'s require minimum m quantities (no's). Each c[i,j] can be computed in time O(m). So the total time for all c[i,j]'s is $O(nm - m^2)$

The total time to compute all the c(i,j)'s and r(i,j)'s is $\sum_{i \le m \le n} (nm - m^2) = O(n^3)$.

0/1 Knapsack Problem.

There are n objects and a knapsack or a bag in which an object i has weight w_i. The knapsack has capacity M. If an object i is placed into the knapsack then a profit of $p_i x_i$ is obtained. Here the objective is to fill the knapsack so that it maximizes the profit obtained and so that sum of weights of objects should not exceed knapsack bag capacity M.

i.e; Maximize $\sum_{i=1}^{n} p_i x_i$ and

Subject to the constraint $\sum_{i=1}^{n} w_i x_i \le M$.

where $1 \le i \le n$

In this 0/1 knapsack problem, $x_i = 0$ or 1. It is not possible to place the fraction (part) of an object into the knapsack. This means, an object is either completly placed into the knapsack or not.

Purging rule or Dominance rule :-

If we have an object pair (P_j, w_j) and (P_k, w_k) such that $(P_j \le P_k)$ and $(w_j \ge w_k)$ then (P_j, w_j) can be eliminated.

This purging rule is also known as Dominance rule.

In purging rule, dominated tuples get purged. i.e; remove the object which has less profit and more weight.

Ex:- (3,5) and (5,4)

$P_j = 3$ $w_j = 5$ $P_k = 5$ $w_k = 4$

$P_j \leq P_k$ and $w_j \geq w_k$.

$3 \leq 5$ and $5 \geq 4$ → True.

So we can eliminate $(P_j, w_j) = (3,5)$

Ex:- Solve 0/1 knapsack instance $M=6$ and $n=3$

Let P_i and w_i are as follows.

$(P_1\ P_2\ P_3) = (1,2,5)$ and $(w_1, w_2, w_3) = (2,3,4)$

We have to build the sequence of decisions S^0, S^1, S^2 and S^3

$S^0 = \{0,0\}$ Initially no object is added. So profit is 0 and weight is 0.

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

$S_1^0 = \{(1,2)\}$

$S^1 = S^0 \cup S_1^0 = \{(0,0)\} \cup \{(1,2)\} = \{(0,0),(1,2)\}$

Next $S_1^1 = \{$ Select next (P,w) pair and add it with $S^1\}$

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

$S^2 = S^1 \cup S_1^1 = \{(0,0)(1,2)\} \cup \{(2,3),(3,5)\}$

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

$S_1^2 = \{$ ~~next~~ Select next (P,w) pair and add it with $S^2\}$

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

$S_1^2 = \{(5,4),(6,6),(7,7),(8,9)\}$

$S^3 = S^2 \cup S_1^2 = \{(0,0),(1,2),(2,3),(3,5)\} \cup \{(5,4),(6,6),(7,7),(8,9)\}$

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

Consider the pairs (7,7) and (8,9). the weights 7 and 9 greater than the capacity of the knapsack M = 6, So. We can remove these pairs from S^3.

$S^3 = \{(0,0), (1,2), (2,3), (3,5), (5,7), (6,6)\}$

By applying the purging rule, the pair (3,5) can be removed.

Consider (3,5) and (5,7) pairs.

(P_j, w_j) and (P_k, w_k)

$(P_j \leq P_k)$ and $(w_j \geq w_k)$

$(3 \leq 5)$ and $(5 \geq 7) \Rightarrow$ True.

So we eliminate pair $(P_j, w_j) = (3,5)$ pair from S^3

$S^3 = \{(0,0), (1,2), (2,3), (5,7), (6,6)\}$

M = 6, we will find the tuple, that contains weight M or near to it.

$(6,6) \in S^3$

check whether $(6,6) \in S^2$

$(6,6) \notin S^2 \quad \therefore x_3 = 1$

$(6-P_3, 6-w_3) = (6-5, 6-4) \in (1,2) \in S$

Next we have to check whether (1,2) or not. $(1,2) \in S^1$, $\therefore x_2 = 0$.

$(1,2) \in S^1$

Next we check whether $(1,2) \in S^0$ or

$(1,2) \notin S^0 \quad \therefore x_1 = 1$.

So optimal solution is (x_1, x_2, x_3)

$= (1,0,1)$

Maximum profit $= \sum_{i=1}^{3} p_i x_i$

$= p_1x_1 + p_2x_2 + p_3x_3 = 1\times1 + 2\times0 + 5\times1$

$= \underline{6}$

Maximum weight $= \sum_{i=1}^{3} w_i x_i$

$= 2\times1 + 3\times0 + 4\times1 = \underline{6}$

Algorithm DKP(P, w, n, m)

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

for i := 1 to n-1 do

{ $S_1^{i-1} := \{(P,w) \mid (P-P_i, W-w_i) \in S^{i-1}$ and $W \leq m\}$;

$S^i := MergePurge(S^{i-1}, S_1^{i-1})$;

}

$(PX, WX) :=$ last pair in S^{n-1};

$(PY, WY) := (P' + P_n, W' + w_n)$

where W' is the largest W in any pair in S^{n-1} such that $W' + w_n \leq m$;

// Trace back for $x_n, x_{n-1} \cdots x_1$.

if $(PX > PY)$ then $x_n := 0$;

else $x_n := 1$;

TraceBackFor($x_{n-1}, \cdots x_1$);

}

Time complexity is $O(nW)$ where n is the number of objects and W is Knapsack capacity.

//

Traveling Salesperson Problem

consider the graph $G = (V, E)$, and assume any vertex as a source.

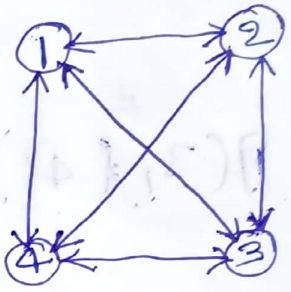
We have to visit all other vertices of a graph with shortest distance or cost.

After visiting all the nodes in a graph, we have to return back to source vertex.

$$cost[i,j] = \begin{cases} 0 & \text{if } i = j \\ c_{ij} & \text{if } (i,j) \in G \\ \infty & \text{if } (i,j) \notin G \end{cases}$$

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

Adjacency matrix



Directed graph

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

i is the source vertex. [or starting vertex]
j is the destination vertex.
S is the set of remaining vertices to be visited.
c(i,j) — cost of the edge<i, j>

If a graph consists four vertices then it will be solved in four stages.

$|S| = 0$

$g(2, \phi) = 5$

$g(3, \phi) = 6$

$g(4, \phi) = 8$

$|S| = 1$

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

$g(2, \{4\}) =$

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

$g(3, \{4\}) =$

$|S| = 1$

$g[2, \{3\}] = \min\{ c_{23} + g(3, \{3\} - 3) \}$

$= \min\{ 9 + g(3, \phi) \}$

$= \min\{ 9 + 6 \} = 15$

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

$= \min\{ c_{24} + g(4, \phi) \}$

$= 10 + 8 = 18$

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

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

$= 13 + 5 = 18$

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

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

$= 12 + 8 = 20$

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

$= \min\{ c_{42} + g(2, \phi) \}$

$= 8 + 5 = 13$

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

$= \min\{ c_{43} + g(3, \phi) \}$

$= 9 + 6 = 15$

$|S| = 2$

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

$c_{24} + g(4, \{3,4\} - 4\}$

$= \min\{ 9 + g(3, \{4\})$

$10 + g(4, \{3\}) \}$

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

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

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

$= \min\begin{cases} 13+18 \\ 12+13 \end{cases} = \min\begin{cases} 31 \\ 25 \end{cases} = 25$

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

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

$= \min\begin{cases} 18+15 \\ 9+18 \end{cases} = \min\begin{cases} 33 \\ 27 \end{cases}$

$= 27$

$|S| = 3$

$g(1,\{2,3,4\})$

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

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

$= \min\begin{cases} 10+25 \\ 15+25 \\ 20+23 \end{cases} = \min\begin{cases} 35 \\ 40 \\ 43 \end{cases} = 35$

Next we need to determine shortest distance path.

We have to start at vertex 1.

$1 \to 2 \to 4 \to 3$

Time complexity is $O(n^2 2^n)$