Optimality is used that means any subpath of shortest path is shortest path blw end nodes. Divide paths from i node to j node for every intermediate node, say 'k'. Then there arises 2 cases

i) path going from i to j via k.

ii) path which is not going via k. Select only shortest path from 2 cases.

Step 3 : The shortest path can be computed using bottom up computation method. following is recursion method.

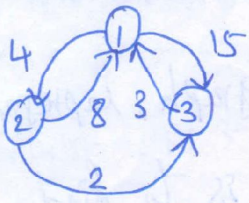
Initially : $A^0 = w[i,j]$

next computations:

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

where $1 \le k \le n$

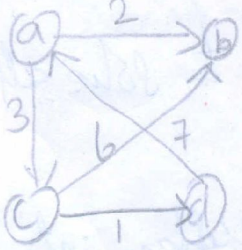
eg:- compute all pair shortest path for following graph.




Algorithm

floyd's shortest path wt[1..n, 1..n]
{
A ← wt // copy weighted matrix to matrix A
for k ← 1 to n do
{
for j ← 1 to n do
{
A[i,j] ← min[A[i,j], A[i,k] + A[k,j]]
}
}
}

Solution:-




$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix} \end{array}$$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 15 \\ 8 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} \end{array}$$

$\min(\infty, (3+4))$ → compute A

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ x & 0 & 2 \end{bmatrix} \end{array}$$

$\min(15, (4+2))$ → compute A

$A^2 =$

| | 1 | 2 | 3 |
|---|---|---|---|
| 1 | 0 | 4 | 6 |
| 2 | 5 | 0 | 2 |
| 3 | 3 | 7 | 0 |

$\xrightarrow{min} (8, (3+2))$

$A^3$ gives shortest path distance blw all pairs

Analysis:-

$$C(n) = \sum_{k=1}^{n} \sum_{j=1}^{n} \sum_{i=1}^{n} i = \sum_{k=1}^{n} \sum_{j=1}^{n} (n-1+1)$$

$$= \sum_{k=1}^{n} \sum_{j=1}^{n} n$$

$$= \sum_{k=1}^{n} n^2$$

$$= n^3$$

time complexity is $O(n^3)$

## Optimal Binary Search Tree (OBST):-

Suppose - search - word - dictionary. every word - looking - becomes time consuming process. To perform - more efficiently - build - binary search tree - key element. again - make - binary search tree efficient by arranging frequently used word nearer to root - < freq words away from root. Such - BST - make - task more simplified as well as efficient. This is called OBST.

## Problem Description:-

Let $\{a_1, a_2, \ldots a_n\}$ be set of identifiers such that $a_1 < a_2 < a_3$. Let $P(i)$ be probability with which we can search for $a_i$ & $q(i)$ be probability of searching element x such that $a_i < x < a_{i+1}$ & $0 \le i \le n$. In other words $P(i)$ probability of successful search & $q(i)$

$1 \le i \le n$ $1 \le j \le n+1$ Then obtain tree with min cost. Such tree with optimum cost is called OBST.

To solve this problem, following steps:-

**Step 1 :**

Notations used

Let,

$T_{ij} = OBST(a_{i+1}, \ldots, a_j)$

$C_{ij}$ denotes the cost $(T_{ij})$

$W_{ij}$ is the weight of each $T_{ij}$

$T_{on}$ is final tree obtained

$T_{oo}$ is empty

$T_{i,i+1}$ is single-node tree that has element $a_{i+1}$

during the computation root values are computed & $r_{ij}$ stores the root value of $T_{ij}$

**Step 2 :**

The OBST can be build using principle of optimality.

Consider the process of creating OBST.

- Let $T_{on}$ be an OBST for elements $a_1 < a_2 < \ldots < a_n$. & let L & R be its left subtree & right subtree.

Suppose that root of $T_{on}$ is $a_k$, for some k.

- Then elements in left subtree L are $a_1, a_2, \ldots, a_{k-1}$

" " right " R are $a_{k+1}, a_{k+2}, \ldots, a_n$

- The cost of computing $T_{on}$ can be given as

we can find another tree L' for the same elements, with the property $C(L') < C(L)$

Let T' be tree with root $a_k$, left subtree L' & right subtree R. Then

$$C(T') = C(L') + C(R) + w$$

ie $$C(T') < C(L) + C(R) + w$$

ie $$< C(T_{on})$$

- That means ; T' is optimal than Ton. This contradicts the fact that Ton is OBST. ∴ L must be optimal for its elements.
- In the same manner we can obtain optimal tree for R.
- Thus we can obtain OBST by building optimal subtrees. This ultimately shows that optimal BST follows Principle of optimality.

Step 3:- we will apply following formula for computing each $c_{eq}$

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

$$w(i,j) = w[i,j-1] + p[j] + q[j];$$

$$r[i,j] = k$$

example:-

Consider $n=4$ & $(q_1, q_2, q_3, q_4) = (do, if, int, while)$. The values for p's & q's are given as

$w_{i,i+1} = q_i + q_{(i+1)} + p_{(i+1)}$

$r_{i,i+1} = i+1$

$c_{i,i+1} = q_i + q_{(i+1)} + p_{(i+1)}$

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

$r_{i,j} = k$

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

we will construct tables for values of $w$, $c$ & $r$.

Let $i = 0$

$w_{00} = q_0 = 2$

when $i=1$

$w_{11} = 3$

when $i=2$

$w_{22} = 1$

when $i = 3$

$w_{33}$

when $i = 4$

$w_{44} = q_4 = 1$

when $i=0$ & $j-i = 1$ then $w_{01} = q_0 + q_1 + p_1$

$= 2 + 3 + 3$

$w_{01} = 8$

when $i=1$ & $j-i = 2$ $w_{12} = q_1 + q_2 + p_2$

$= 3 + 1 + 3$

$w_{12} = 7$

when $i=2$ & $j-i = 3$ $w_{23} = q_2 + q_3 + p_3$

$= 1 + 1 + 1$

$w_{23} = 3$

when $i=3$ & $j-i = 4$ $w_{34} = q_3 + q_4 + p_4$

$= 1 + 1 + 1$

$w_{34} = 3$

when $i=0$ & $j-i = 2$ $w_{ij} = w_{i,j-1} + p_j + q_j$

$$w_{234} = w_{23} + p_4 + q_4$$
$$= 3 + 1 + 1$$
$$w_{24} = 5$$

when $i=0$ & $j-i=3$ then

$$w_{03} = w_{02} + p_3 + q_3$$
$$= 12 + 1 + 1$$
$$= 14$$

when $i=1$ & $j-i=3$ then

$$w_{14} = w_{13} + q_4 + p_4$$
$$= 9 + 1 + 1$$
$$= 11$$

when $i=0$ & $j-i=4$ then

$$w_{04} = w_{03} + q_4 + p_4$$
$$= 14 + 1 + 1$$
$$= 16$$

The table for w can be represented as →p

| j−i ↓ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | $w_{00}=2$ | $w_{11}=3$ | $w_{22}=1$ | $w_{33}=1$ | $w_{44}=1$ |
| 1 | $w_{01}=8$ | $w_{12}=7$ | $w_{23}=3$ | $w_{34}=3$ | |
| 2 | $w_{02}=12$ | $w_{13}=9$ | $w_{24}=5$ | | |
| 3 | $w_{03}=14$ | $w_{14}=11$ | | | |
| 4 | $w_{04}=16$ | | | | |

we will now compute for c & r.

As $c_{i,i} = 0$ & $r_{i,i} = 0$

$c_{00}=0$ $c_{11}=0$ $c_{22}=0$ $c_{33}=0$ $c_{44}=0$

$r_{00}=0$ $r_{11}=0$ $r_{22}=0$ $r_{33}=0$ $r_{44}=0$

Similarly, $c_{i,i+1} = q_i + q_{(i+1)} + p_{(i+1)}$ & $r_{i,i+1}$

∴ when $i=0$

$$c_{01} = q_0 + q_1 + p_1$$
$$= 2 + 3 + 3$$

$= 1+1+1 = 3$ & $r_{23} = 3$

when $i=1$ $\quad C_{34} = q_3 + P_4 + q_4$

$= 1+1+1 = 3$ & $r_{34} = 4$

now we will compute $c_{ij}$ & $r_{ij}$ for $j-1 \geq 2$.

As $\quad C_{i,j} = i \leq k \leq j \min \{ C_{(i,k-1)} + C_{kj} \} + w_{ij}$

hence we will find $k$.

→ for $C_{02}$ we have $i=0$ & $j=2$. for $r_{i,j-1}$ to $r_{i+1,j}$ i.e for $r_{01}$ to $r_{1,2}$. we will compute minimum value of $c_{ij}$.

→ Let $r_{01} = 1$ & $r_{12} = 2$. Then we will assume value of $k=1$ & will compute $c_{ij}$. Similarly with $k=2$ we will compute $c_{ij}$ & will pick up minimum value of $c_{ij}$ only.

Let us compute $c_{ij}$ with following formula,

$$c_{ij} = c_{i,k-1} + c_{kj}$$

for $k=1$, $i=0$, $j=2$

$$C_{02} = C_{00} + C_{12}$$
$$= 0 + 7 = 7$$

for $k=2$, $i=0$, $j=2$

$$C_{02} = C_{01} + C_{22}$$
$$= 8 + 0 = 8$$

0 2 1
0 2 2

0 3 1
0 3 2
3
0 4 1
0 4 2
3
4
1 3 k=2
3
1 4 2
3
4
2 4 3
4

→ ①

→ ②

from equ 1 & 2 we can select min value of $C_{02}$ is 7. that means $k=1$ gives us min value of $c_{i,j}$.

Hence $\quad r_{ij} = r_{02} = k = 1$

now $\quad C_{02} = \min \{ C_{(i,k-1)} + C_{kj} \} + w_{ij}$

$= 7 + w_{02}$

$= 7 + 12$ ...