Date: __.__.__

④ DYNAMIC PROGRAMMING

① Greedy Method } Optimization
② Dynamic Programming } (min / max)

Greedy ⇒ Pre defined procedure is followed to get optimal result. The procedure is known to be optimal.

Dynamic Programming ⇒ All possible solutions are calculated and among them, the best (optimal) solution is picked. This is time consuming wrt greedy method

↓

recursion / iteration (mostly)

Principle of optimality : A problem can be solved by taking a sequence of decisions to get the optimal solution.

↓

Greedy : Decission is taken ONE time.

Dynamic Programming : Decission is taken at every stage.

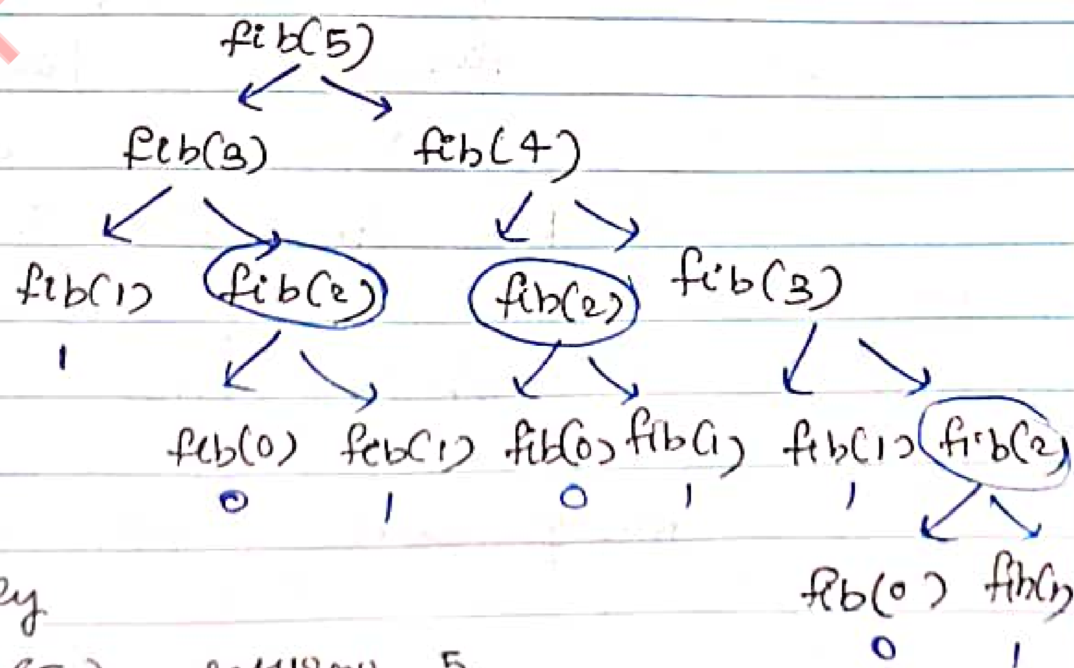
Fibonacci series function

$$fib(n) = \begin{cases} 0 & ; \forall n = 0 \\ 1 & ; \forall n = 1 \\ fib(n-2) + fib(n-1) & ; \forall n > 1 \end{cases}$$

C code :

```c
int fib (int n)
{
        if (n <=1) return n;
        else return fib (n-2)+ fib(n-1);
}
```

Fibonacci series : 0, 1, 1, 2, 3, 5, 8, 13 . . .
0 1 2 3 4 5 6 7



Finally
fib (5) returns 5.

Recurrence relation

$$T(n) = T(n-1) + T(n-2) + 1$$

approximately

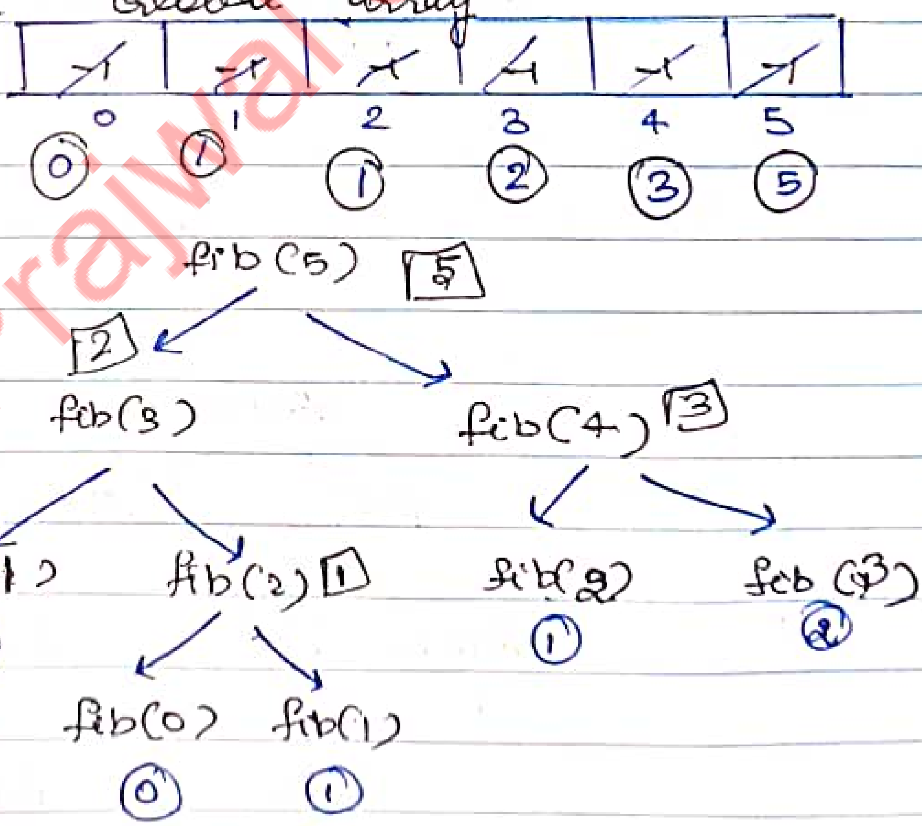
$$T(n) = 2T(n-1) + 1$$

By master's theorem $\rightarrow$ $\boxed{O(2^n)}$

Problem : same function is being called for the same n again and again.

↓

Solution : Global array



known as

No. of calls $f(n) = n+1$ $\boxed{O(n)}$

MEMOIZATION $\rightarrow$ reduces time complexity

Top Down approach

Iterative Approach

```
int fib (int n)
{
    if (n <= 1) return n;
    F[0] = 0; F[1] = 1;

    for (int i = 2; i <= n; i++)
        F[i] = F[i-2] + F[i-1];

    return F[n];
}
```

fib(5):

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

(F(5) = 5 returned)

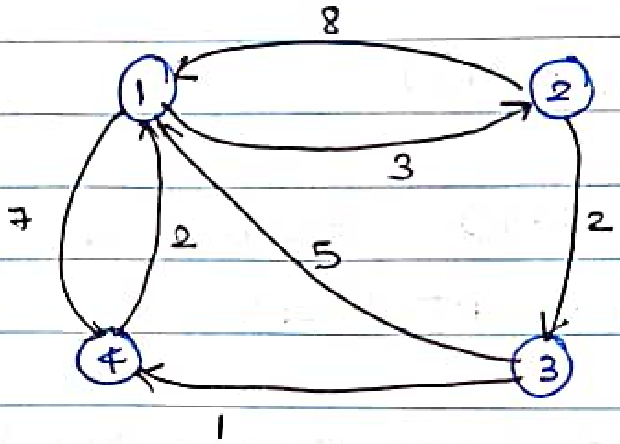
Bottom-Up Approach : start from 0 to 5
Top-Down Approach
↳ start from top (here 5) and then keep going below (here till 0)

Preference : Iterative approach → Dynamic Programming

apsara

Date: ___ . ___ . ___

ALL PAIRS SHORTEST PATH → (Floyd warshall)



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

$1 \to 2, 3, 4$ $\quad$ $2 \to 1, 3, 4$ $\quad$ $3 \to 1, 2, 4$ $\quad$ $4 \to 1, 2, 3$

can be done using Dijkstra's algorithm
$O(n^2)$ on n vertices one by one → $O(n^3)$ time complexity

taking ① as the intermediate vertex

$$A^1 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & \infty & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & \infty & 0 \end{array}$$

Filling $A_1[2,3]$

$A_0[2,3] = 2$ ✓ min

$A_0[2,1] + A_0[1,3]$
$= 8 + \infty = \infty$

So $A_1[2,3] = 2$

For $A_1[2,4]$ $\quad$ $A^0[2,4] = \infty$

$A^0[2,1] + A^0[1,4] = 8 + 7 = 15$ ✓ min

Similarly filling all

$\because A^0[3,2] \quad A^0[3,1] + A^0[1,2]$

$\infty > 5 + 3$

$A^0[3,4] \quad A^0[3,1] + A^0[1,4]$

$1 < 5 \quad 7$

$A^0[4,2] \quad A^0[4,1] + A^0[1,2]$

$\infty > 2 \quad 3$

$A^0[4,3] \quad A^0[4,1] + A^0[1,3]$

$\infty \not< 2 \quad \infty$

Taking ② as intermediate matrix

$$A^2 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 7 \\ 2 & 8 & 0 & 2 & 15 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

Taking ③ as intermediate matrix

$$A^3 : \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 7 & 0 & 2 & 3 \\ 3 & 5 & 8 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

Taking 4 as intermediate matrix

$$A^4 = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 3 & 5 & 6 \\ 2 & 5 & 0 & 2 & 3 \\ 3 & 3 & 6 & 0 & 1 \\ 4 & 2 & 5 & 7 & 0 \end{array}$$

Formula :

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

code :

```
for (k=1; k<=n; k++)
{   for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
        {
            A[i,j] = min(A[i,j], A[i,k]
                                  + A[k,j]);
        }
    }
}
```

Time complexity ↪ $\boxed{O(n^3)}$

MATRIX CHAIN MULTIPLICATION

$$A_1 \cdot A_2 \cdot A_3 \cdot A_4$$
$$5\times4 \quad 4\times6 \quad 6\times2 \quad 2\times7$$

Condition for matrix multiplication

no. of columns of first matrix = no. of rows of second matrix

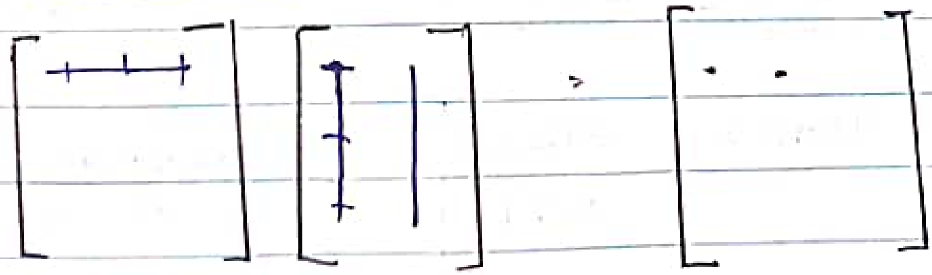
eg

$$\underset{5\times 4}{A} \quad \underset{4\times 3}{B} \quad \text{possible ✓}$$

same

AB possible does not imply BA is necessarily possible

$$A \cdot B = C$$
$$5\times4 \quad 4\times3 \quad 5\times3$$



Time complexity = $5 \times 4 \times 3$

= 60 multiplications to be performed

A_1 A_2 A_3 A_4

$\underbrace{5}_{d_0}\times\underbrace{4\quad 4}_{d_1}\times\underbrace{6\quad 6}_{d_2}\times\underbrace{2\quad 2}_{d_3}\times\underbrace{7}_{d_4}$

↓

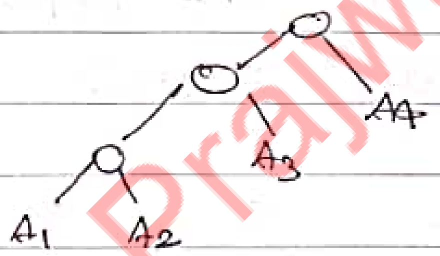
pair wise multiplication must be performed.

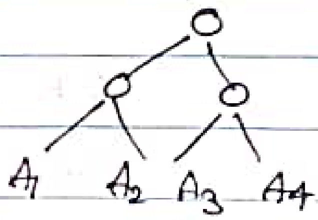
↓

which pair and order should be selected such that the total cost of multiplication is minimum

parenthesis : $((A_1\ A_2)\ A_3)\ A_4$

$(A_1\ A_2)(A_3\ A_4)$





no. of possibilities : no. of trees (T)

$$\boxed{T(n) = \frac{2nC_n}{n+1}} \rightarrow \text{(catalan numbers)}$$

$$T(3) = \frac{6C_3}{4} = \frac{20}{4} = 5$$

↳ 5 trees

Date: __.__.____

solution using Dynamic Programming :

(i)

m	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

S	1	2	3	4
1		1	1	3
2			2	3
3				3
4				

1 at a time m[1,1] m[2,2] m[3,3] m[4,4]

A_1 A_2 A_3 A_4

2 at a time m[1,2] :

$A_1 . A_2$ (5x4) (4x6) => cost is 120

m[2,3]

$A_2 . A_3$ (4x6) (6x2) => 48

m[3,4]

$A_3 . A_4$ (6x2) (2x7) => 84

3 at a time m[1,3]

$A_1 . A_2 . A_3$

$A_1 (A_2 \; A_3)$
5x4 4x6 6x2

m[1,1] + m[2,3] + 5x4x2

= 0 + 48 + 40 = $\underline{88}$ min

$(A_1 \; A_2) \; A_3$
5x4 4x6 6x2

m[1,2] + m[3,3] + 5x6x2 = 120+0+60 = $\underline{180}$

apsara

Date: ___.___.___

$m[2,4]$

$A_2 \; (A_3 \;\; A_4)$ $\qquad$ $(A_2 \;\; A_3) \; A_4$

$4\times6 \;\; 6\times2 \;\; 2\times7$ $\qquad$ $4\times6 \;\; 6\times2 \;\; 2\times7$

$m[2,2] + m[3,4]$ $\qquad$ $m[2,3] + m[4,4]$

$+ \; 4\times6\times7$ $\qquad$ $+ \; 4\times2\times7$

$= 0 + 84 + 168$ $\qquad$ $= 0 + 48 + 56$

$= 252$ $\quad < \quad$ $= 104$ <u>min</u>

4 at a time

↓

1) $m[1,1] + m[2,4] + (5\times4\times7)$

$= 0 + 104 + 140 = 244$

2) $m[1,2] + m[3,4] + (5\times6\times7)$

$= 120 + 84 + 210 = 414$

3) $m[1,3] + m[4,4] + (5\times2\times7)$

$= 88 + 0 + 70 = 158$

$\min(158, 414, 244) = \boxed{158}$

Formula :

$$\boxed{m[i,j] = \{ \min [i,k] + m[k+1,j] + d_{i-1} * d_k * d_j \}}$$

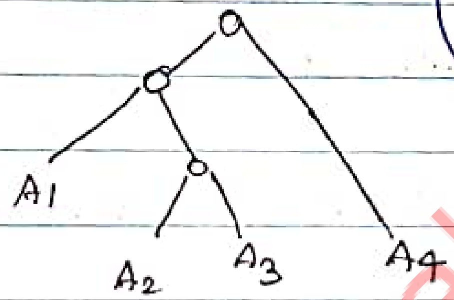
Grouping using S matrix

$A_1 \ A_2 \ A_3 \ A_4$ $\qquad S[1,4] = 3$, split @ 3

$(A_1 \ A_2 \ A_3)(A_4)$ $\qquad$ Now $S[1,3] = 1$ split @ 1

$\boxed{((A_1)(A_2 \ A_3))(A_4)}$ $\qquad$ Now $S[2,3] = 2$ split @ 2

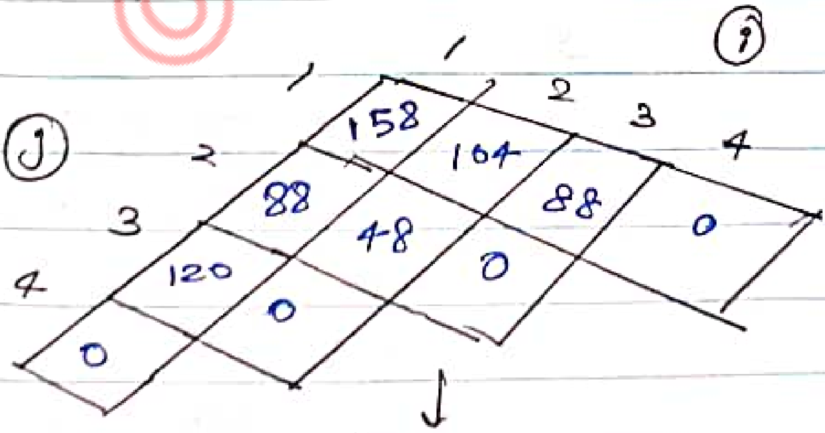
done



(order of multiplication)

space complexity $\frac{n(n-1)}{2} = O(n^2)$

Time complexity $\frac{n(n-1)}{2} \cdot n \rightarrow O(n^3)$



(Another way of drawing the table)

Program for matrix cham multiplication

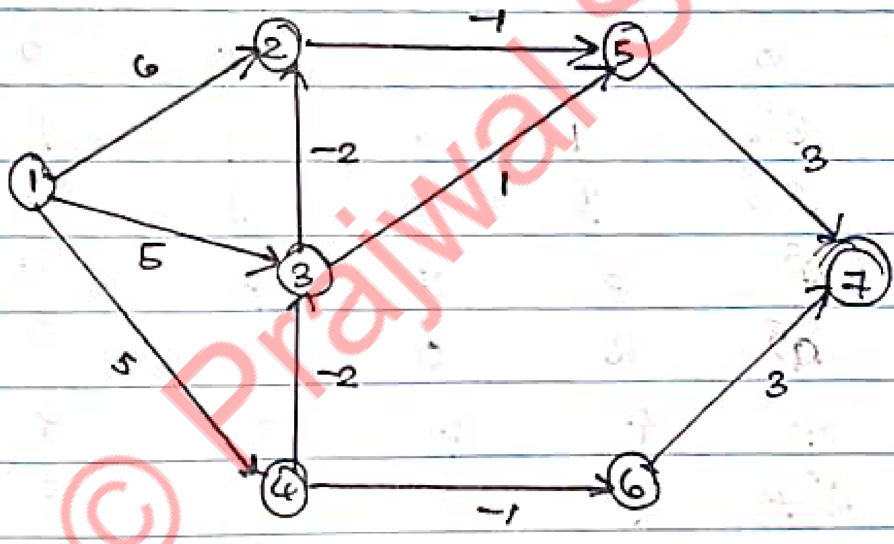
p	5	4	6	2	7	array of dimensions
	0	1	2	3	4	

```
main()
{
    int n = 5;
    int p[] = {5, 4, 6, 2, 7};
    int m[5][5] = {0};
    int s[5][5] = {0};
    int j, mm, q;

    for (int d = 1; d < n-1; d++)
    {
        for (int i = 1; i < n-d; i++)
        {
            j = i+d;
            mm = 32767;
            for (int k = 1; k <= j-1; k++)
            {
                q = m[i][k] + m[k+1][j]
                    + p[i-1] * p[k] * p[j];
                if (q < mm)
                {
                    mm = q;
                    s[i][j] = k;
                }
        }
```

```
            m[i][j] = min ;
        }
    }

    cout << m[1][n-1];
}
```

SINGLE SOURCE SHORTEST PATH ALGORITHM

BELMAN - FORD



$|V| = n = 7$

Perform relaxation on edges $|V| - 1$ times

Here, $|V| - 1 = 7 - 1 = 6$ times

Relaxation :

```
if ( d[u] + c(u, v) < d[v] )
        d[v] = d[u] + c(u, v);
```

Edge List

(1,2)	(1,3)	(1,4)	(2,5)	(3,2)	(3,5)	(4,3)
$\boxed{6}$ ①	$\boxed{5}$ ②	$\boxed{5}$ ③	$\boxed{-1}$ ④	$\boxed{-2}$ ⑤	$\boxed{1}$ ⑥	$\boxed{-2}$ ⑦

(4,6)	(5,7)	(6,7)
$\boxed{-1}$ ⑧	$\boxed{3}$ ⑨	$\boxed{3}$ ⑩

10 edges present

vertex (u) Distance (d) → initially d_0, finally d_{n-1}

v	d_0	d_1	d_2	d_3	d_4	d_5	d_6
1	0	0	0	0	0	0	0
2	∞	~~6~~ 3	1	1	1	1	1
3	∞	~~5~~ 3	3	3	3	3	3
4	∞	~~∞~~ 5	5	5	5	5	5
5	∞	~~∞~~ 5	2	0	0	0	0
6	∞	4	4	4	4	4	4
7	∞	~~8~~ 7	5	3	3	3	3

shortest path

Time complexity

$$O(|E|(|V|-1)) = \boxed{O(|V||E|)}$$

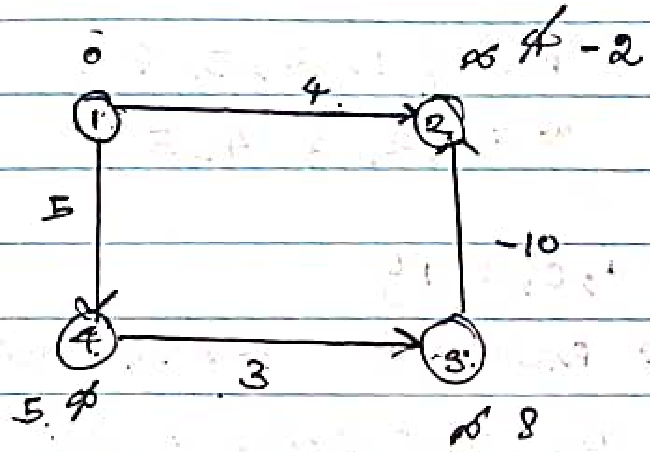
$|V| = |E| = n$

$$= \boxed{O(n^2)}$$

complete graph

$$|E| = \frac{n(n-1)}{2} \rightarrow O(|V||E|) = \boxed{O(n^3)}$$

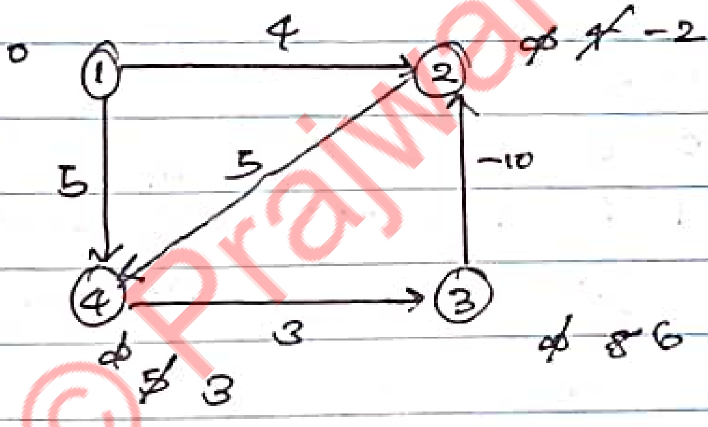
Another example :



1 →	0
2 →	(-2)
3 →	8
4 →	5

Edge list : (3,2) (4,3) (1,4) (1,2)

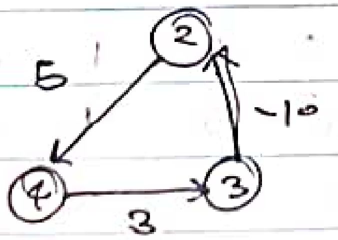
Drawback of Bellman-Ford :



Even after n-1 iterations, at each step, relaxation occurs.

Edge list : (3,2) (4,3) (1,4) (1,2) (2,4)

Reason :



Total weight = 5+3-10 = -2

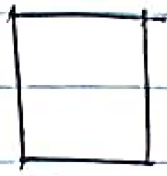
negative edge cycle, path keeps reducing till $-\infty$. Bellman Ford fails.

But in n^{th} iteration, if any change → -ve edge cycle can be detected.

0/1 KNAPSACK PROBLEM

$m = 8$ $\quad\quad$ $p = \{1, 2, 5, 6\}$

$n = 4$ $\quad\quad$ $w = \{2, 3, 4, 5\}$



$m=8$

$x = \{1, 0, 0, 1\}$

0 → excluded $\quad$ 1 → included

objects are indivisible : NO fractions

objective : $\max \sum p_i x_i$ $\quad$ max profit

$\sum w_i x_i \leq m$

No. of possible solns

0 0 0 0

1 1 1 1 . . . . 2^m solutions

(not all are feasible)

Tabulation method :

			0	1	2	3	4	5	6	7	8
P	w	0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

(max profit)

apsara

Date: __.__.____

$$\boxed{V[i,w] = \max\{ V[i-1,w] , \; P[i] + V[i-1, w-w[i]] \}}$$

sequence of objects chosen :

$\{x_1, x_2, x_3, x_4\} = \{0, 1, 0, 1\}$

$\Downarrow$

x_2 and x_4 are included

Set Method : (P, w) profit ↙ ↪ weight

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

$S^0_1 = \{(1,2)\}$

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

↓ (2,3) added

$S^1_1 = \{(2,3), (3,5)\}$

⇓ merge

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

↓ (5,4) added

$S^2_1 = \{(5,4), (6,6), (7,7), \cancel{(8,9)}\}$ exceeds capacity

⇓ merge

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

dominance rule

if weight ↑ profit ↑ but 3,5 → 5,4

w ↑ P ↓ ⇒ wrong ⇒ eliminate pair with lesser profit

apsara

Date: ___.___.___

$S_1^3 = \{ (6,5), (7,7), (8,8), \cancel{(11,9)},$
$\cancel{(12,11)}, \cancel{(13,12)} \}$

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

final set

object 4 : $(8,8)$ belongs to $S^4 \in S^4, \notin S^3$.
$x_4 = 1$, x_4 included
$(8-6, 8-5) = (2,3)$

object 3 : $(2,3) \in S^3, \in S^2$
$x_3 = 0$, x_3 excluded

object 2 : $(2,3) \in S^2, \notin S^1$
$x_2 = 1$, x_2 included
$(2-2, 3-3) = (0,0)$

object 1 : $(0,0) \in S^1 \in S^0$
$x_1 = 0$, x_1 excluded.

$x = \{0, 1, 0, 1\}$

x_2 & x_4 included, max profit

apsara

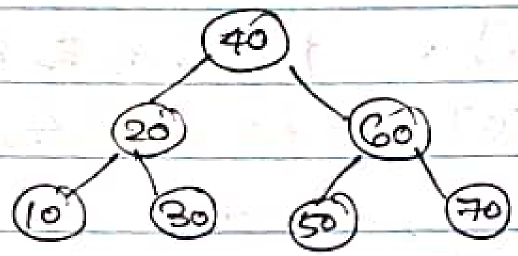
Date: __.__.__

Program for 0/1 knapsack :

```
main()
{
    int p[5] = {0, 1, 2, 5, 6};
    int wt[5] = {0, 2, 3, 4, 5};
    int m = 8, n=4;
    int k[5][9];

    for (int i=0; i<=n; i++)
    {
        for (int w=0; w<=m; w++)
        {
            if (i==0|| w==0)
                K[i][w] =0;
            else if (wt[i] <= w)
                K[i][w] = max( K[i-1][w],
                    P[i] + K[i-1][w - wt[i]]);
            else
                K[i][w] = K[i-1][w];
        }
    }
    cout << k[n][w];
while( i>0 && j>0)
{   if ( k[i][j]==K[i-1][j]) { cout<<i<< "=0"<<endl;
                                i--; }
    else { cout<< i << "=1" <<endl; i--;
           s -= wt[i] };
};
```

OPTIMAL BINARY SEARCH TREE

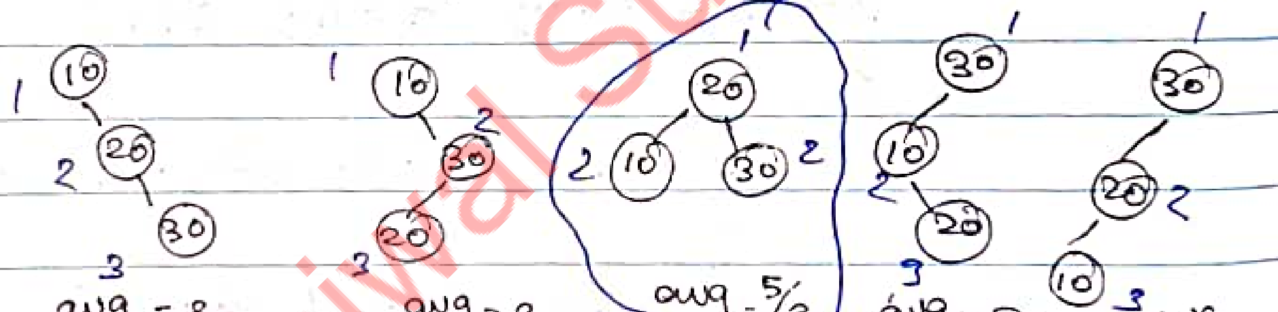
Keys : 10, 20, 30, 40, 50, 60, 70



O(log n) time to search

Keys : 10, 20, 30

balanced BST



avg = 2   avg = 2   avg = 5/3   avg = 2   avg = 2

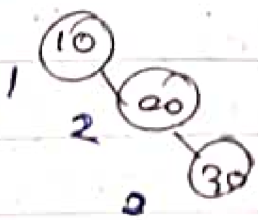
no. of Binary search trees (BSTs)

$$= \boxed{\frac{{}^{2n}C_n}{n+1}} \Rightarrow \text{Here } n=3, \quad \frac{{}^{6}C_3}{4} = \frac{20}{4} = 5$$

Frequency of searching

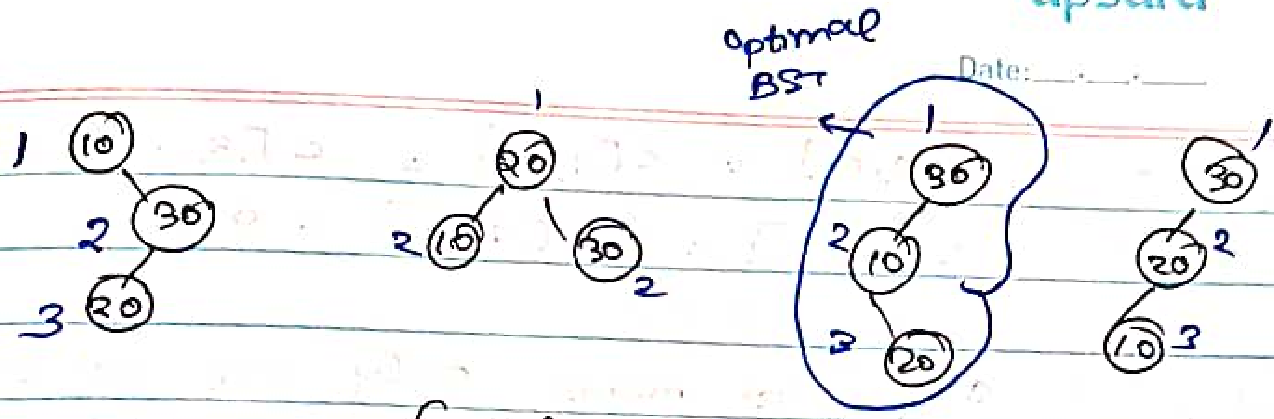
10	20	30
3	2	5

Now



$(1\times3) + (2\times2) + (3\times5)$

$= 3 + 4 + 15 = 22$



$(1\times3)+(2\times5)+(3\times2) = 19$

$(1\times2)+(2\times3)+(2\times5) = 18$

$(1\times5)+(2\times3)+(3\times2) = 17$

least cost

$(1\times5)+(2\times2)+(3\times3) = 18$

Even though it is not height balanced, it is the optimal BST.

Finding optimal BST using concept of Dynamic Programming

	1	2	3	4
keys	10	20	30	40
Frequency	4	2	6	3

cost table

i \ j	0	1	2	3	4
0	0	4 $\boxed{1}$	8 $\boxed{1}$	20 $\boxed{2}$	26 $\boxed{3}$
1		0	2 $\boxed{2}$	10 $\boxed{3}$	16 $\boxed{3}$
2			0	6 $\boxed{3}$	12 $\boxed{3}$
3				0	3 $\boxed{4}$
4					0

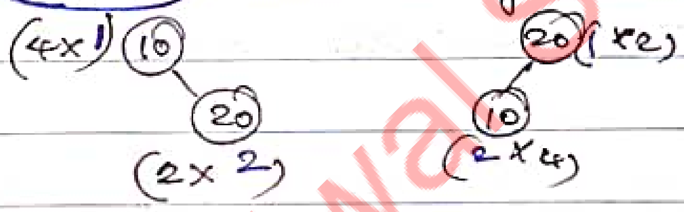
j-i=0 $c[0,0] = c[1,1] = c[2,2] = c[3,3] = c[4,4] = 0$

j-i=1 $c[0,1] \Rightarrow$ means only one key is present

key k1 → cost = ? cost = 4

$c[0,1] = 4, \quad c[1,2] = 2, \quad c[2,3] = 6, \quad c[3,4] = 3$

j-i=2 $c[0,2]$ considering k1 and k2



cost = 8 cost = 10

⇓

min

cost min = 8 with root ⇒ k1

$$\boxed{w(i,j) = \sum_{n=i+1}^{j} f(n)}$$

$c[0,2] = c[0,0] + c[1,2] + w[0,2]$

$= 0 + 2 + (4+2) = 8$ ✓ min

(OR)

$c[0,1] + c[2,2] + w[0,2]$

$4 + 0 + (4+2) = 10$

$c[1,3]$ keys k_2 and k_3

key 2 : $c[1,1] + c[2,3] + w[1,3)$ 20 30

$= 0 + 6 + (2+6) = 14$

key 3 : $c[1,2] + c[3,3] + w(1,3)$ 30 20

$= 2 + 0 + (2+6) = 10 \rightarrow$ min.

$c[2,4]$ keys k_3 and k_4

key 3 : $c[2,2] + c[3,4] + w(2,4)$

$= 0 + 3 + (6+3) = 12 \longrightarrow$ min

key 4 : $c[2,3] + c[4,4] + w(3,4)$

$= 6 + 0 + (6+3) = 15$

$j-i=3$ $c[0,3]$

$k=1$ $c[0,0] + c[1,3] = 0 + 10 = 10$

$k=2$ $c[0,1] + c[2,3] = 4 + 6 = 10$

$k=3$ $c[0,2] + c[3,3] = 8 + 0 = 8 \rightarrow$ min

$8 + w(0,3) = 8 + (4+2+6) = 20$

$c[1,4]$

$k=2$ $c[1,1] + c[2,4] = 0 + 12 = 12$

$k=3$ $c[1,2] + c[3,4] = 2 + 3 = 5 \longrightarrow$ min

$k=4$ $c[1,3] + c[4,4] = 10 + 0 = 10$

$5 + w(1,4) = 5 + (2+6+3) = 16$

$j-i=4$ $\quad c[0,4]$

$k=1 \quad c[0,0]+c[1,4] = 0+16 = 16$

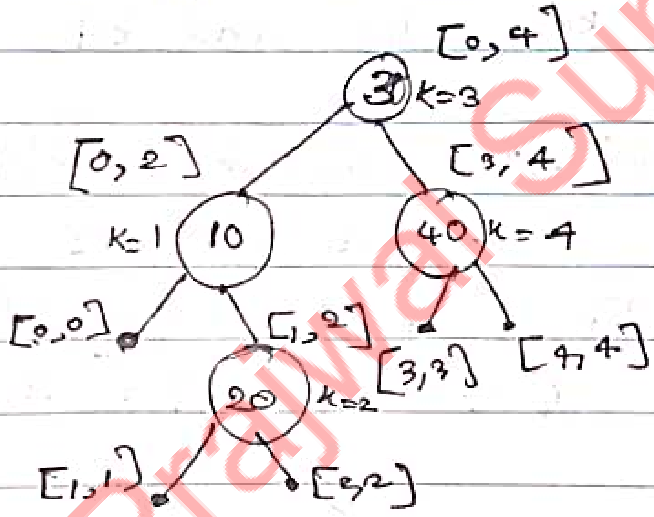
$k=2 \quad c[0,1]+c[2,4] = 4+12 = 16$

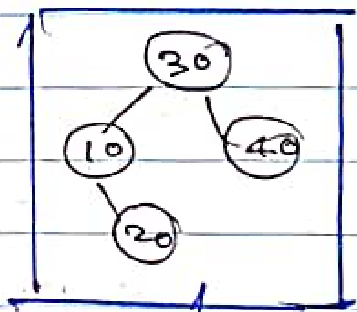
$k=3 \quad c[0,2]+c[3,4] = 8+3 = 11 \rightarrow \min$

$k=4 \quad c[0,3]+c[4,4] = 20+0 = 20$

$11 + w(0,4) = 11 + (4+2+6+3) = 26$

Generating Tree





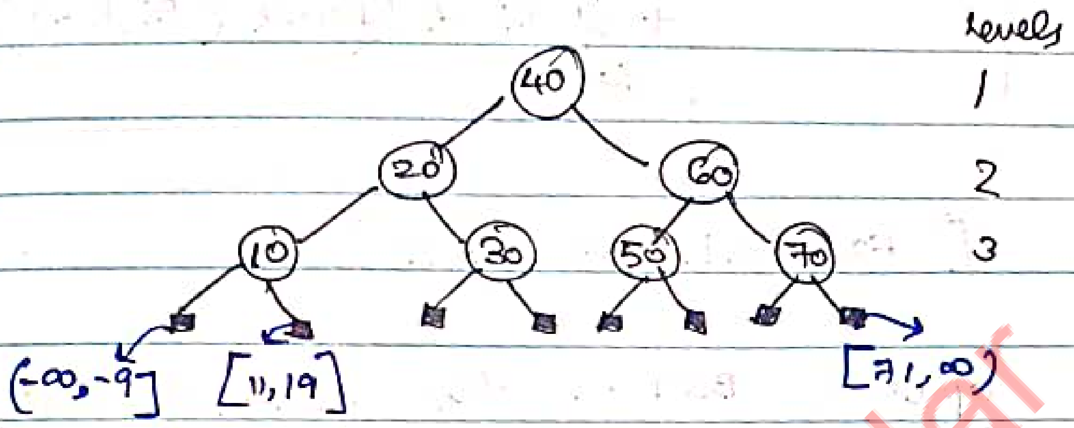
is the required Optimal BST.

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

(formula)

$$\boxed{w(i,j) = \sum_{n=i+1}^{j} f(n)}$$

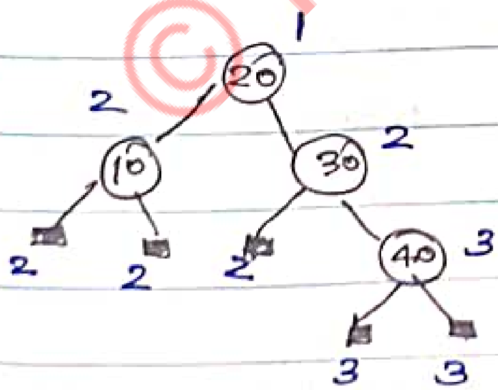
successful and unsuccessful search



■ → square, dummy node, representing unsuccessful search

Probabilities

keys		10		20		30		40
p_i		0.1 p_1		0.2 p_2		0.1 p_3		0.2 p_4
q_i	0.1 q_0		0.05 q_1		0.15 q_2		0.05 q_3	0.05 q_4



$(1\times 0.2) + (2\times 0.1) + (2\times 0.1) + (3\times 0.2)$ } success

$+ \{ (2\times 0.1) + (2\times 0.05) + (2\times 0.15) + (3\times 0.05) + (3\times 0.05)$ unsuccessful

$= 0.2 + 0.2 + 0.2 + 0.6 + = 1.2$

$0.2 + 0.1 + 0.3 + 0.15 + 0.15 + 0.4 = 2.1$

cost

$$\boxed{\text{cost}[0,n] = \sum_{i=1}^{n} p_i * \text{level}(a_i) + \sum_{i=0}^{n} q_i * (\text{level}(E_i) - 1)}$$

↓ cost formula

[For existing BST]

optimal BST → has minimal cost

Raw approach : Draw all BSTs, find all costs and extract minimum cost BST

↓

But this is very time consuming

Eg $n = 4$ $\quad \frac{{}^{2n}C_n}{n+1} = \frac{{}^{8}C_4}{5} = \frac{70}{5} = 14$

14 trees can't be drawn.

↓ solution

Use of dynamic programming.

cost function

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

apsara

Date: __.__.____

eg $c[0,3]$

$$c[0,3] = \min \begin{Bmatrix} c[0,0] + c[1,3], \\ c[0,1] + c[2,3], \\ c[0,2] + c[3,3] \end{Bmatrix} + w(0,3)$$

$$c[1,3] = \min \begin{Bmatrix} c[1,1] + c[2,3], \\ c[1,2] + c[3,3] \end{Bmatrix} + w(1,3)$$

To calculate larger values, smaller values are needed

↳ use Bottom-Up Approach
[Tabular Approach]

$j-i=0$	$c[0,0]$	$c[1,1]$	$c[2,2]$	$c[3,3]$
$j-i=1$	$c[0,1]$	$c[1,2]$	$c[2,3]$	
$j-i=2$	$c[0,2]$	$c[1,3]$		
$j-i=3$	$c[0,3]$			

$$w[0,2] = q_0 + p_1 + q_1 + p_2 + q_2$$

$$w[0,3] = \underbrace{q_0 + p_1 + q_1 + p_2 + q_2}_{w[0,2]} + p_3 + q_3$$

↓

$$\boxed{w[i,j] = w[i, j-1] + p_j + q_j}$$

weight function

Form an optimal BST :-

	0	1	2	3	4
Keys		10	20	30	40
p_i		3	3	1	1
q_i	2	3	1	1	1
$p_i + q_i$		6	4	2	2

weight Table

w

	0	1	2	3	4
0	2	8	12	14	16
1		3	7	9	11
2			1	3	5
3				1	3
4					1

cost Table

c

	0	1	2	3	4
0	0	8	19	25	32
1		0	7	12	19
2			0	3	8
3				0	3
4					0

Root Table

R

	0	1	2	3	4
0	0	1	1	2	2
1		0	2	2	2
2			0	3	3 $\cancel{4}$
3				0	4
4					0

$j-i=0$ $\quad c[i,i] = 0 \ , \ R[i,i] = 0$

$w[i,i] = q_i$

$w[0,0] = 2 \quad w[1,1] = 3 \quad w[2,2] = 1 \quad w[3,3] = 1 \quad w[4,4] = 1$

$j-i=1$

$c[0,1]$

$K=1$ $\quad c[0,0]+c[1,1] = 0+0=0 \rightarrow \min$

$c[0,1] = 0+w(0,1) = 0+8 = 8$

$R[0,1] = 1$

$c[1,2]$

$K=2$ $\quad c[1,1]+c[2,2] = 0+0=0 \rightarrow \min$

$C[1,2] = 0+w(1,2) = 0+7 = 7$

$R[1,2] = 2$

$c[2,3]$

$k=3$ $\quad c[2,2]+c[3,3] = 0+0=0 \rightarrow \min$

$c[2,3] = 0+w(2,3) = 0+3 = 3$

$R[2,3] = 3$

$c[3,4]$

$K=4$ $\quad c[3,3]+c[4,4] = 0+0=0 \rightarrow \min$

$c[3,4] = 0+w(3,4) = 0+3 = 3$

$R[3,4] = 4$

$j-i=2$

$c[0,2]$

$k=1$ $\quad c[0,0]+c[1,2] = 0+7=7 \rightarrow \min$

$k=2$ $\quad c[0,1]+c[2,2] = 8+0 = 8$

$c[0,2] = 7+w(0,2) = 7+12 = 19$

$R[0,2] = 1$

$c[1,3]$

$k=2$ $c[1,1] + c[2,3] = 0+3 = 3 \rightarrow \min$

$k=3$ $c[1,2] + c[3,3] = 7+0 = 7$

$c[1,3] = 3 + w(1,3) = 3+9 = 12$

$R[1,3] = 2$

$c[2,4]$

$k=3$ $c[2,2] + c[3,4] = 0+3 = 3$

$k=4$ $c[2,3] + c[4,4] = 3+0 = 3$ $\rightarrow$ min

$c[2,4] = 3 + w(2,4) = 3+5 = 8$

$R[2,4] = 3/4$

$j-i=3$

$c[0,3]$

$k=1$ $c[0,0] + c[1,3] = 0+12 = 12$

$k=2$ $c[0,1] + c[2,3] = 8+3 = 11 \rightarrow \min$

$k=3$ $c[0,2] + c[3,3] = 19+0 = 19$

$c[0,3] = 11 + w(0,3) = 11+14 = 25$

$R[0,3] = 2$

$c[1,4]$

$k=2$ $c[1,1] + c[2,4] = 0+8 = 8 \rightarrow \min$

$k=3$ $c[1,2] + c[3,4] = 7+3 = 10$

$k=4$ $c[1,3] + c[4,4] = 12+0 = 12$

$c[1,4] = 8 + w(1,4) = 8+11 = 19$

$R[1,4] = 2$

$j-i=4$ $c[0,4]$

$k=1$ $c[0,0] + c[1,4] = 0 + 19 = 19$

$k=2$ $c[0,1] + c[2,4] = 8 + 8 = 16 \longrightarrow$ min

$k=3$ $c[0,2] + c[3,4] = 19 + 3 = 22$

$k=4$ $c[0,3] + c[4,4] = 25 + 0 = 25$

$c[0,4] = 16 + w(0,4) = 16 + 16 = 32$

$R[0,4] = 2$

Optimum cost

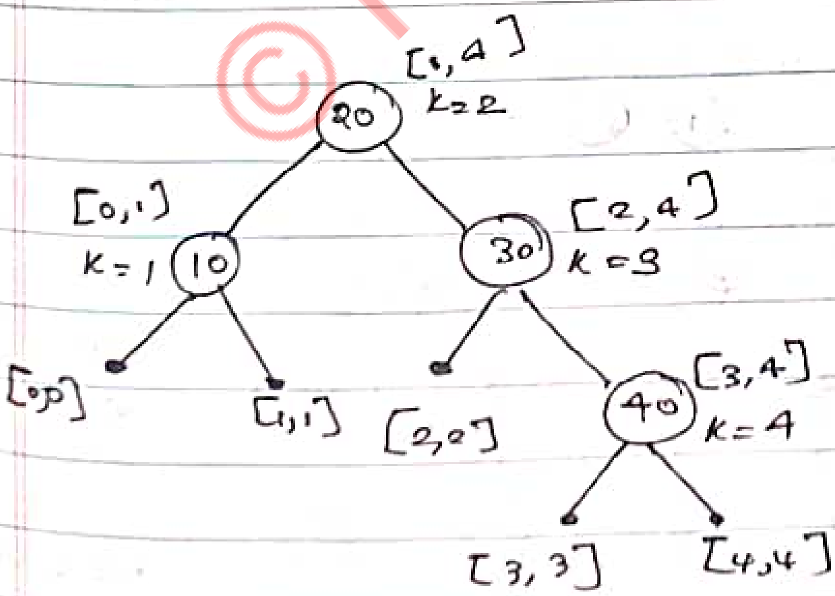
Total frequency $= \sum p_i + \sum q_i = \sum(p_i) + \sum(q_i)$

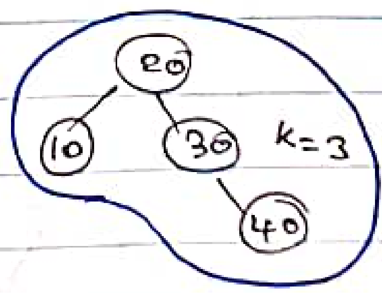
$= 8 + 8 = 16$

$$\text{cost} = \frac{c[0,4]}{16} = \frac{32}{16} = 2$$

2 ← min cost

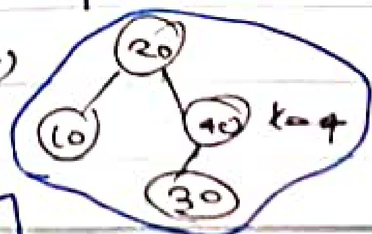
Generating Tree :





is an optimal BST

(or)

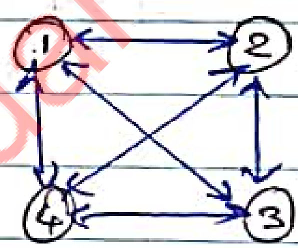


[same cost]

TRAVELLING SALESMAN PROBLEM

$$\boxed{g(i,S) = \min_{k \in S} \{ c_{ik} + g(k, S - \{k\}) \}}$$

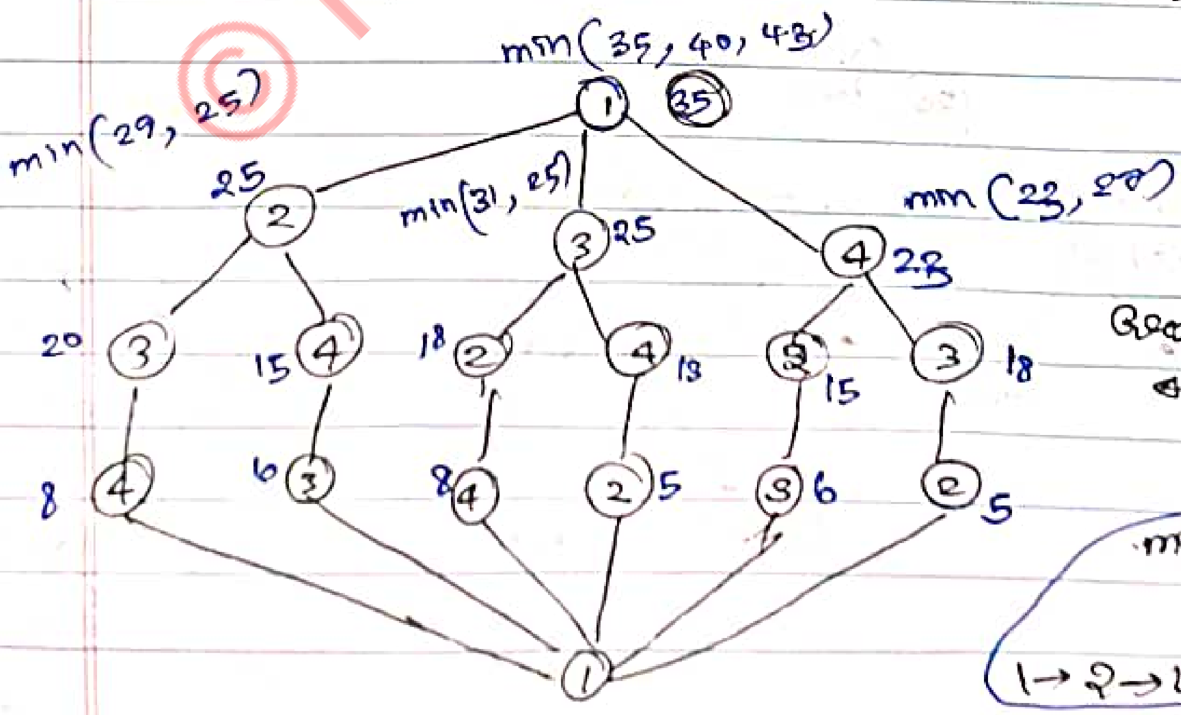
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \end{array}$$



start from some vertex, travel to all other vertices exactly once, and then return to the starting point.

↓

cost of travel must be minimum



Graph ←

minimum cost = 35
1 → 2 → 4 → 3 → 1

when no vertex is remaining

$g(2, \phi) = 5$

$g(3, \phi) = 6$

$g(4, \phi) = 8$

when one vertex is remaining

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

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

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

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

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

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

when two vertices are remaining

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

$$= \min \begin{bmatrix} 9 + 20, \\ 10 + 15 \end{bmatrix} = \min(29, 25) = 25$$

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

$$= \min \begin{bmatrix} 13 + 18, \\ 12 + 13 \end{bmatrix} = \min(31, 25) = 25$$

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

$$= \min \begin{bmatrix} 8 + 15, \\ 9 + 18 \end{bmatrix} = \min(23, 27) = 23$$

When three vertices are remaining

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

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

$$= \min \begin{bmatrix} 10 + 25, \\ 15 + 25, \\ 20 + 23 \end{bmatrix} = \min(35, 40, 43) = 35$$

clearly minimum cost = 35

route: keep locating minimum path

@ 1 min. came at 2

@ 2 min came @ 4

@ 4 go to 3

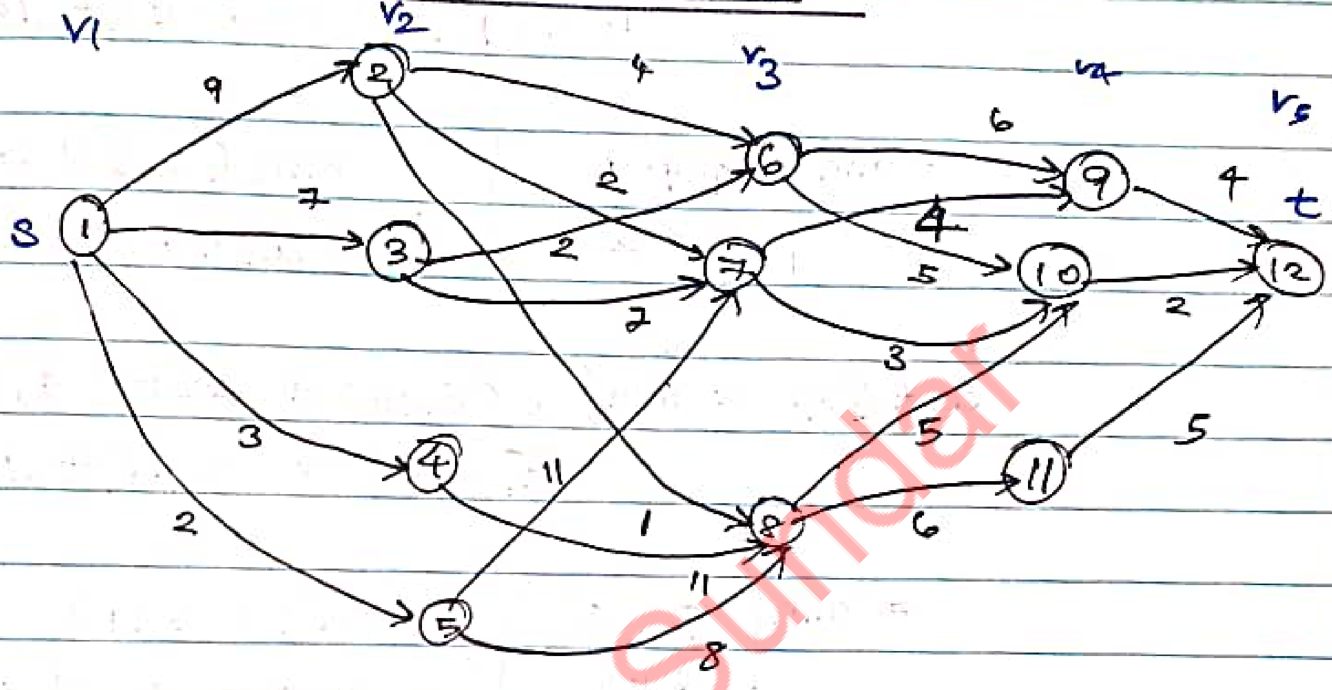
@ 3 go to 1

$\Downarrow$

$$\boxed{\begin{array}{c} 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1 \\ \text{min cost} = 35 \end{array}}$$

Date: __.__.__

MULTI - STAGE GRAPH



v	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

stage 5

stage vertex

cost (5, 12) = 0

stage 4

cost (4, 9) = 4

cost (4, 10) = 2

cost (4, 11) = 5

stage 3

$$\text{cost}(3,6) = \min\begin{bmatrix} c(6,9) + \text{cost}(4,9), \\ c(6,10) + \text{cost}(4,10) \end{bmatrix}$$

$$= \min\begin{bmatrix} 6+4, \\ 5+2 \end{bmatrix} \therefore \min(10, 7) = 7$$

cost = 7, d = 10

$$cost(3,7) = \min\begin{bmatrix} c(7,9) + cost(4,9) \\ c(7,10) + cost(4,10) \end{bmatrix}$$

$$= \min\begin{bmatrix} 4+4, \\ 3+2 \end{bmatrix} = \min(8,5) = 5$$

$cost = 5,\ d = 10$

$$cost(3,8) = \min\begin{bmatrix} c(8,10) + cost(4,10), \\ c(8,11) + cost(4,11) \end{bmatrix}$$

$$= \min\begin{bmatrix} 5+2, \\ 6+5 \end{bmatrix} = \min(7,11)$$

$cost = 7,\ d = 10$

stage 2

$$cost(2,2) = \min\begin{bmatrix} c(2,6) + cost(3,6), \\ c(2,7) + cost(3,7) \\ c(2,8) + cost(3,8) \end{bmatrix}$$

$$= \min\begin{bmatrix} 4+7, \\ 2+5, \\ 1+7 \end{bmatrix} = \min(11,7,8) = 7$$

$cost = 7$

$d = 7$

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

$$= \min\begin{bmatrix} 2+7 \\ 7+5 \end{bmatrix} = \min(9,12) = 9$$

$cost = 9,\ d = 6$

$cost(2,4) = c(4,8) + cost(3,8)$

$= 11 + 7 = 18$

$cost = 18, \quad d = 8$

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

$$= min\begin{bmatrix} 11 + 5 \\ 8 + 7 \end{bmatrix} = min(16, 15) = 5$$

$cost = 15, \quad d = 8$

stage 1

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

$$= min\begin{bmatrix} 9 + 7 \\ 7 + 9 \\ 3 + 18 \\ 2 + 15 \end{bmatrix} = min(16, 16, 21, 17)$$

$= 16$

$cost = 16, \quad d = 2 \text{ or } 3$

Formula

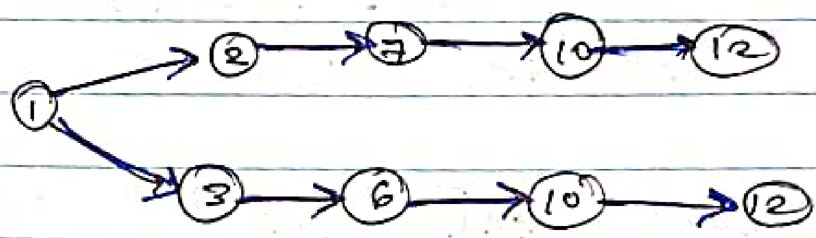
$$\boxed{cost(i,j) = min[c(j,l) + cost(i+1, l)]}$$

stage vertex

$\langle j, l \rangle \in E, \quad l \in V_{i+1}$

Date: __.__.__

Now locating path :



[least cost paths]

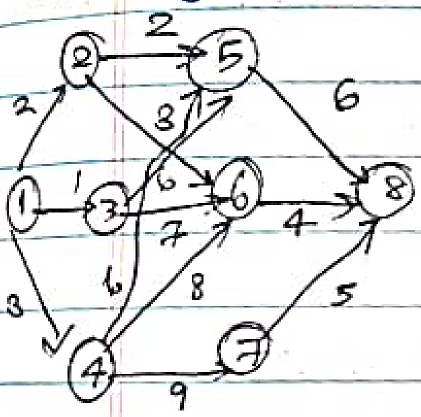
d(1,1) = 2		(OR)	d(1,1) = 3
d(2, 2) = 7			d(2,3) = 6
d(3, 7) = 10			d(3,6) = 10
d(4, 10) = 12			d(4,10) = 12

Program :

```
main()
{
    int stages = 4, min;
    int n = 8;
    int cost[9], d[9], path[9];
    int c[9][9] = { {0, 0, 0, 0, 0, 0, 0, 0, 0},
                    {0, 0, 2, 1, 3, 0, 0, 0, 0},
                    {0, 0, 0, 0, 0, 2, 3, 0, 0}
                    ....    };

    cost[n] = 0;

                 →
```

```
for ( int i = n-1; i >= 1; i++)
{
    min = 32767;
    for ( k = i+1; k <= n; k++)
    {
        if ( c[i][k] != 0 &&
             c[i][k] + c[k] < min)
        {
            min = c[i][k] + c[k];
            d[i] = k;
        }
    }
    cost[i] = min;
}
```

Graph used in program:



$1+2+3+\cdots+n$

$= \frac{n(n+1)}{2}$

$= O(n^2)$

```
P[1] = 1;    P[stages] = n;

for (i = 2; i < stages; i++)
    P[i] = P[d[i-1]];
}
```

	0	1	2	3	4	5	6	7	8
cost		9	7	11	12	6	4	5	0
d		2	6	6	5	8	8	8	
path		1	2	6	8				

$1 \to 2 \to 6 \to 8$

apsara

Date: __.__.__

LONGEST COMMON SUBSEQUENCE

string 1 : a b c d e f g h i j

string 2 : c d g i

cdgi is the longest common subsequence

a b c d e f g h i j

egi → e c d g i

a b c d e f g h i j

cdgi → e c d g i

a b d a c e

bace → b a b c e

a b d a c e

abce → b a b c e

Recursive Algorithm :

```
int lcs (i, j)
{
    if ( A[i] == '\0' || B[j] == '\0' )
        return 0;
    else if ( A[i] == B[j] )
        return 1+ lcs (i+1, j+1);
```

else

return max(LCS(i+1, j), LCS(i, j+1));

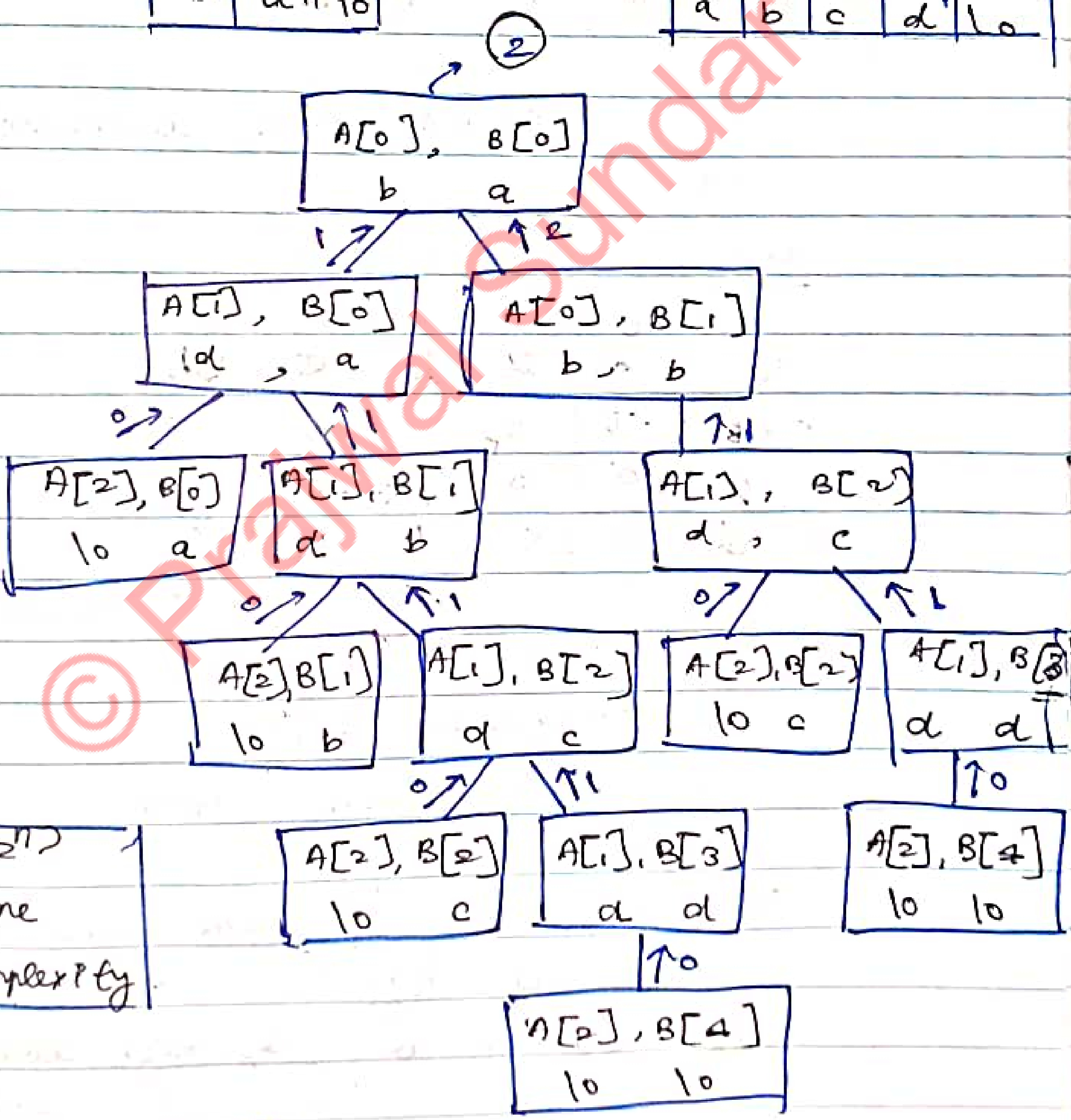
}

A

0	1	2
b	d	\0

B

0	1	2	3	4
a	b	c	d	\0



$O(2^n)$
time
complexity

Finally 2 is returned.

Memoisation :

		0	1	2	3	4
		a	b	c	d	lo
0	b	2	2			•
1	d	1	1	1	1	•
2	lo	0	0	0	•	0

(storing result)

$\boxed{O(mn)}$ time complexity

Dynamic Programming :

		0	1	2	3	4
		~~a~~	a	b	c	d
0	~~#~~	0	0	0	0	0
1	b	0	0	1	1	1
2	d	0	0	1	1	2

(answer)

↑b ↑d

Subsequence bd.

$\boxed{O(mn)}$ time

If not matching, take max of top and left.

Else add 1 to top left diagonal element.

str1 : s t o n e

str2 : l o n g e s t

		0	1	2	3	4	5	6	7
			l	o	n	g	e	s	t
0		0	0	0	0	0	0	0	0
1	s	0	0	0	0	0	0	1	0
2	t	0 ←	0	0	0	0	01	1	2
3	o	0	0	1	1	1	1	1	2
4	n	0	0	1	2 ←	2	2	2	2
5	e	0	0	1	2	2	3 ←	3 ←	3

↑o ↑n ↑e

one is the LCS.

length = 3

RELIABILITY DESIGN

setup a system

D1 — D2 — D3 — D4 (series)

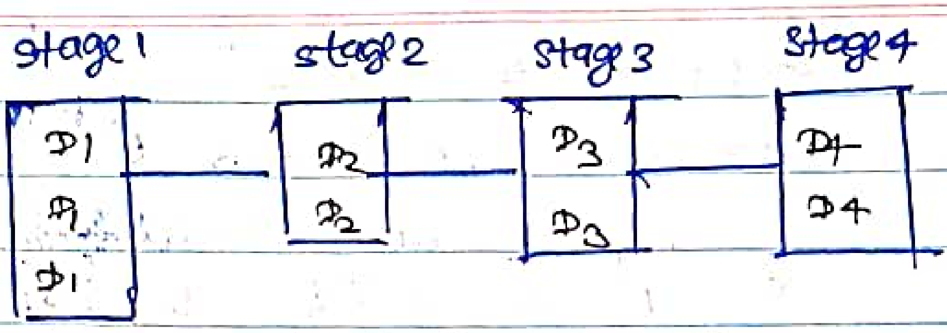
c_1 c_2 c_3 c_4

r_1 r_2 r_3 r_4

0.9 0.9 0.9 0.9

Total reliability = $\prod r_i = (0.9)^4 = 0.6561$

Target : maximum reliability



$\Rightarrow$ all in series overall

$\Rightarrow$ each in parallel within

$r_1 = 0.9$, $\quad 1 - r_1 = 1 - 0.9 = 0.1$

All failed = $(1-r_1)^3 = (0.1)^3 = 0.001$

At least one success = $1 - 0.001 = \underbrace{0.999}_{\text{increased!}}$

How many devices of each type to buy within cost constraints such that the overall reliability of the system is maximum?

Di	ci	ri	ui
D1	30	0.9	2
D2	15	0.8	3
D3	20	0.5	3

$c = 105$

$\Sigma c_i = c_1 + c_2 + c_3$

$= 30 + 15 + 20 = 65$

remaining $c - \Sigma c_i = 105 - 65 = 40$

$1 + \left\lfloor \frac{c - \Sigma c_i}{c_i} \right\rfloor = 1 + \left\lfloor \frac{40}{30} \right\rfloor = 1 + 1 = 2$ copies

$1 + \left\lfloor \frac{40}{15} \right\rfloor = 1 + 2 = 3$ copies

$1 + \left\lfloor \frac{40}{20} \right\rfloor = 1 + 2 = 3$ copies

(R, c)

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

consider D_1 $\quad \times 0.9 \quad + 30$

1 copy $\quad S_1^1 = \{ \boxed{(0.9, 30)} \}$ $\qquad 1 - (1 - 0.9)^2 = 1 - (0.1)^2 = 1 - 0.01 = 0.99$

$m_1 = 1$

2 copies $\quad S_2^1 = \{(0.99, 60)\}$

consider D_2

1 copy $\quad S^1 = \{ \boxed{(0.9, 30)}, (0.99, 60) \}$

$\downarrow (0.8, 15)$

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

$1 - (1 - 0.8)^2 = 1 - (0.2)^2 = 1 - 0.04 = 0.96$

2 copies $\quad S_2^2 = \{ \boxed{(0.864, 60)}, \cancel{(0.9504, 90)} \}$

$m_2 = 2$

$105 - 90 = 15 < 20$ (cost of 3^{rd} device)

90 cost ×

Date: __.__.__

$$1-(1-0.8)^3 = 1-(0.2)^3 = 1-0.008$$
$$= 0.992$$

3 copies

$$S_3^2 = \{(0.8928, 75)\}$$

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

Reliability ↑ cost ↑ (Dominance Rule)

If not → remove ordered pair with higher cost

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

considering D_3

$$S_1^3 = \{(0.36, 65), (0.432, 80), (0.4466, 95)\}$$

$$1-(0.5)^2 = 1-0.25 = 0.75$$

cost = 40

$D_3 = 2$

2 copies → $\{(0.54, 85), \boxed{(0.648, 100)}, \cancel{(-, 115)}\}$

$$1-(0.5)^3 = 1-0.125 = 0.875$$

3 copies → $\{(0.63, 105), \cancel{(-, 120)}, \cancel{(-, 135)}\}$

apsara

Date: ___.___.___

$S^3 = \{ (0.36, 65), (0.432, 80), (0.44\cancel{64}, 95), (0.54, 85), (0.6\cancel{8}, 105), (0.64, 100) \}$

$S^3 = \{ (0.36, 65), (0.432, 80), (0.54, 85), \underbrace{\boxed{(0.64, 100)}}_{\text{maximum}} \}$

Maximum reliability = 0.64, cost = 100

D_1	D_2	D_3
1	2	2

[no. of copies]