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

$P_1\ w_1$ $P_2\ w_2$

purging rule.

→ Take maximum profit $(6,6) \in S^3$

$\notin S^2$

we consider 3rd object.

$(x_3 = 1)$

$\Rightarrow (6-4)\ (-1,2) \in S^2 \Rightarrow x_2 = 0$

$\in S^1$

$(1,2) \in S^1 \Rightarrow x_1 = 1.$

$\notin S^0$

12/05/2023

* MATRIX CHAIN MULTIPLICATION:

↳ Minimum no. of multiplications should be done.

Eg: A_1 A_2 A_3 A_4

2×4 4×3 3×4 4×2

$2\times3\times4$

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

2×3

$2\times4\times3$ 2×4 2×2

total multiplications

$= 24+24+16$

$= 64$

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

$2\times3\times4 \qquad 3\times2\times4$

$24 \qquad 24$

$2\times3 \quad 3\times2$

$2\times3\times2 = 12$

total multiplications

$= 24+24+12$

$= 60$

C

	1	2	3	4
1	0	–	–	–
2		0		
3			0	
4				0

cost[1,2] → j−i = 1 → no. of combin^n of k.

cost[1,3] → j−i = 2

cost[1,4] → j−i = 3

split numbr

S

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

① $j-i=1$

$C[1,2] = 2\times3\times4 = 24 \quad (k=1)$

$C[2,3] = 4\times4\times3 = 48 \quad (k=2)$

$C[3,4] = 3\times2\times4 = 24 \quad (k=3)$

C	1	2	3	4
1	0	24	48	60
2		0	48	48
3			0	24
4				0

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

$$c[i,j] = \min_{i \le k < j} \{ c[i,k] + c[k+1,j] + d_{i-1} \cdot d_k \cdot d_j \}$$

d = dimension

② $j - i = 2$

$c[1,3] = \min \{ c[1,1] + c[2,3] + d_0 d_1 d_3 \;,\; c[1,2] + c[3,3] + d_0 d_2 d_3 \}$

$1 \le K \le 3$

$K = 1/2$

$A_1 \; A_2 \; A_3$

$2\times4 \;\; 4\times3 \;\; 3\times4$

$d_0 d_1 \;\; d_1 d_2 \;\; d_2 d_3$

$= \min \{ 0 + 48 + 2\times4\times4 \;,\; 24 + 0 + 2\times3\times4 \}$

$= \min \{ 80, 48 \}$

$\therefore K = 2$

$A_2 \; A_3 \; A_4$

$4\times3 \;\; 3\times4 \;\; 4\times2$

$d_1 \;\; d_2 \;\; d_2 d_3 \;\; d_3 d_4$

$c[2,4] = \min \{ c[2,2] + c[3,4] + d_1 d_2 d_4 \;,\; c[2,3] + c[4,4] + d_1 d_3 d_4 \}$

$K = 2/3$

$= \min \{ 0 + 24 + 4\times3\times2 \;,\; 48 + 0 + 4\times4\times2 \}$

$= \min \{ 48, 80 \}$

$K = 2$

$j-i=3$ $k=1, 2, 3$

$$c[1,4] = \min\{c[1,1]+c[2,4]+d_0 d_1 d_4,$$
$$c[1,2]+c[3,4]+d_0 d_2 d_4,$$
$$c[1,3]+c[4,4]+d_0 d_3 d_4\}$$

A_1	A_2	A_3	A_4
2×4	4×3	3×4	4×21
d_0 d_1	d_1 d_2	d_2 d_3	d_3 d_4

$$= \min\{0+48+2\times4\times2,\ 24+24+2\times3\times2,$$
$$48+0+2\times4\times2\}$$

$$= \min\{64, 60, 64\}$$

$k=2$

order: $((A_1 A_2)(A_3 A_4))$

A_1	A_2	A_3	A_4
5×4	4×6	6×2	2×7

$j-i=1$ $\quad 5\times6 \quad 5\times2 \quad 5\times7$

$$c[1,2] = 5\times6\times4 = 120 \quad (k=1)$$
$k=1$

$$c[2,3] = 4\times2\times6 = 48 \quad (k=2)$$
$k=2$

c	1	2	3	4	5	6	7
1	0	120	88				
2		0	48	104			
3			0	84			
4				0			
5					0		

$c[3,4] = 6 \times 7 \times 2 \quad (k=3)$

$= 84$

s	1	2	3	4	5	6	7
1	0	1	1				
2		0	2	3			
3			0	3			
4				0			
5					0		

$j - i = 2$

$A_1 \quad A_2 \quad A_3$

$5 \times 4 \quad 4 \times 6 \quad 6 \times 2$

$d_0 \; d_1 \quad d_1 \; d_2 \quad d_2 \; d_3$

$c[1,3] = \{k = 1, 2\}$

$= \min\{c[1,1] + c[2,3] + d_0 d_1 d_3,$

$c[1,2] + c[3,3] + d_0 d_2 d_3\}$

$= \min\{0 + 48 + 5 \times 4 \times 2,$

$120 + 0 + 5 \times 6 \times 2\}$

48
40

120
6

$= \min\{88, 180\}$ $(k=1)$

$A_2 \quad A_3 \quad A_4$

$d_1 d_2 \quad d_2 d_3 \quad d_3 \; d_4$

$4 \times 6 \quad 6 \times 2 \quad 2 \times 7$

$c[2,4] = \{k = 2, 3\}$

$= \min\{c[2,2] + c[3,4] + d_1 d_2 d_4,$

$c[2,3] + c[4,4] + d_1 d_3 d_4\}$

42
14
168
84
84
52

$= \min\{0 + 84 + 4 \times 6 \times 7, \; 48 + 0 + 4 \times 2 \times 7\}$

48
56
104

$= \min\{252, 104\}$

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

* OPTIMAL BINARY SEARCH TREE:

22/05/2023

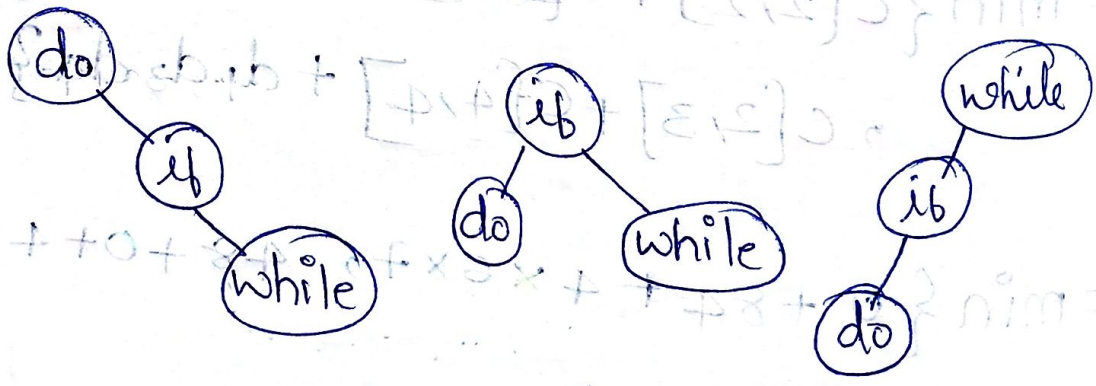
→ Construct a binary tree; where the cost of binary tree is minimum.

* cost of binary tree = cost of left subtree + cost of right subtree.

* This optimal binary search tree is used to efficiently differentiate b/w keywords and identifiers in the compiler.

*→ If a set of keywords are given; we must find out optimal BST whose cost is minimum.

*



i → node

* cost of successful searches = for internal nodes.

Probability(i) * Level at which it is present

= p(i) * level(i)

cost of unsuccessful searches $= q_i * \text{Level}(E_i - 1)$

q_i ↓ Probability of unsuccessful searches.

Level $(E_i - 1)$ → $(i-1)^{th}$ level.

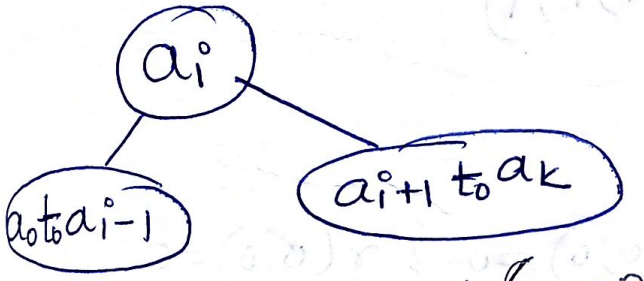
$$\text{total cost} = \sum p_i * \text{level}(E_i) + \sum q_i * (\text{level}(E_i - 1))$$

Ex: $p_1 = \frac{1}{2}$ $p_2 = \frac{1}{4}$, $p_3 = \frac{1}{4}$

$q_0 - q_3 = \frac{1}{8}$.

* Given set of identifiers $a_1 < a_2 < \cdots < a_k$

→ Let $a_i \Rightarrow$ optimal identifier for root.



$\text{cost}(T) = p_i + \text{cost}(\text{left}) + \text{cost}(\text{right})$

→ const (cannot be changed)

↓ ↓

shd be min

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

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

- $w(i,i) = q(i)$, $c(i,i) = 0$,

  $r(i,i) = 0$

P = probability of successful search

q = probability of unsuccessful search

* $n = 4$; (a_1, a_2, a_3, a_4)

$(p_1 \cdots p_4) = (3,3,1,1)$

$(q_0 \cdots q_4) = (2,3,1,1,1)$

* $j - i = 0$

$w(0,0) = q_0 = 2$; $c(0,0) = 0$; $r(0,0) = 0$

$w(1,1) = q_1 = 3$; $c(1,1) = 0$; $r(1,1) = 0$

$w(2,2) = q_2 = 1$; $c(2,2) = 0$; $r(2,2) = 0$

$w(3,3) = q_3 = 1$; $c(3,3) = 0$; $r(3,3) = 0$

$w(4,4) = q_4 = 1$; $c(4,4) = 0$; $r(4,4) = 0$

$w(0,1) = p(1) + q(1) + w(0,0)$

$= 3 + 3 + 2 = 8$

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

$c(0,1) = \min_{k=1} \{ c(0,0) + c(1,1) + w(0,1) \}$

$c(0,1) = 0 + 0 + 8 = 8$

$c(0,1) = 8$

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

$= 3 + 1 + 3 = 7.$

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

$c(1,2) = \min_{k=2} \{ c(1,1) + c(2,2) + w(1,2) \}$

$= 0 + 0 + 7 = 7$

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

$= 1 + 1 + 1 = 3.$

$c(2,3) = \min_{k=3} \{ c(2,2) + c(3,3) + w(2,3) \}$

$= 0 + 0 + 3 = 3.$

$r(2,3) = 3$

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

$= 1 + 1 + 1 = 3.$

$c(3,4) = \min_{k=4} \{ c(3,3) + c(4,4) + w(3,4) \} = 3$

$r(3,4) = 4$

$* \rightarrow j - i = 2$

$\rightarrow$

$W(0,2) = P(2) + q(2) + W(0,1)$

$= 3 + 1 + 8 = 12$

$$c(0,2) = \min_{k=1,2} \{ (c(0,0) + c(1,2) + w(0,2)), (c(0,1) + c(2,2) + w(0,2)) \}$$

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

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

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

$\rightarrow W(1,3) = P(3) + q(3) + w(1,2)$

$= 1 + 1 + 7 = 9$

$$c(1,3) = \min_{k=2,3} \{ (c(1,1) + c(2,3) + w(1,3)), (c(1,2) + c(3,3) + w(1,3)) \}$$

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

$= 12$

$r(1,3) = 2$

$w(2,4) = p(4) + q(4) + w(2,3)$

$= 1 + 1 + 3 = 5$

$c(2,4) = \min_{k=3,4} \{ (c(2,2) + c(3,4) + w(2,4)), (c(2,3) + c(4,4) + w(2,4)) \}$

$= \min\{0 + 3 + 5, 3 + 0 + 5\}$

$= 8$

$r(2,4) = 3$

$j - i = 3$

$w(0,3) = p(3) + q(3) + w(2,2)$

$= 1 + 1 + 12 = 14$

$c(0,3) = \min_{k=1,2,3} \{ (c(0,0) + c(1,3) + w(0,3)), (c(0,1) + c(2,3) + w(0,3)), (c(0,2) + c(3,3) + w(0,3)) \}$

$= \min\{(0 + 12 + 14), (8 + 3 + 14), 19 + 0 + 14\}$

$$c(0,3) = \min\{26, 25, 33\}$$

$$c(0,3) = 25$$

$r = k = 2$

$$\rightarrow w(1,4) = p(4) + q(4) + w(1,3)$$

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

$k = 2, 3, 4$

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

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

$$c(1,4) = 19$$

$j - i = 4$

$$w(0,4) = p(4) + q(4) + w(0,3)$$

$$= 1 + 1 + 14 = 15.$$

$k = 1, 2, 3, 4$

$$c(0,4) = \min\{$$

23/05/2023

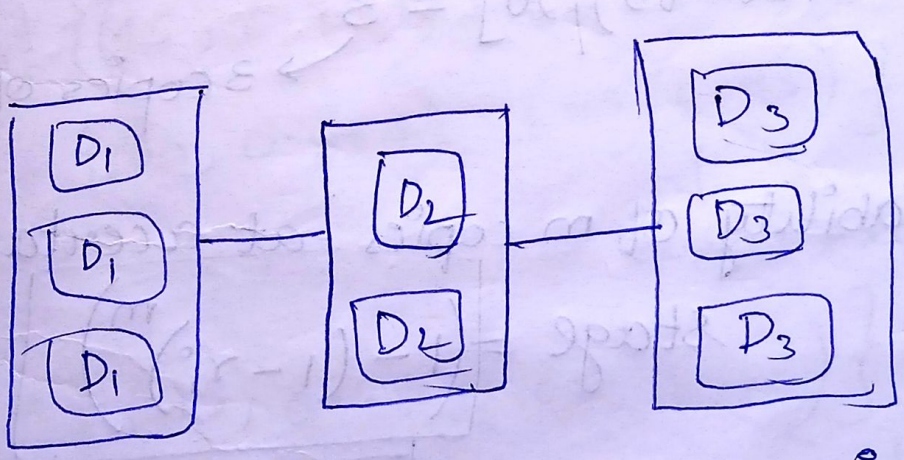
→ Reliability Design problem:

* Given devices $D_1, D_2, \ldots, D_n$.

for any device D_i → Cost(c_i)
→ reliability (r_i).

*→ Connecting the no. of devices in such a way that; the reliability of the system is improved.

* Using Dynamic Programming; we construct different stages where in each stage; there are 'm' copies of each device D_i.



* Keeping the cost constraint in mind; we shall select a ~~stage~~ connection which has maximum reliability; but minimum cost.

* $C = 105$ $(c_1, c_2, c_3) = (30, 15, 20)$

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

$$u_i = \left[\left(C + c_i - \sum_{j}^{n} c_j \right) / c_i \right]$$

$$u_1 = \left[(C + c_1 - (65)) / 30 \right]$$

$$u_1 = \left[(105 + 30 - 65) / 30 \right] = (2).33.$$

→ 2 copies of D_1

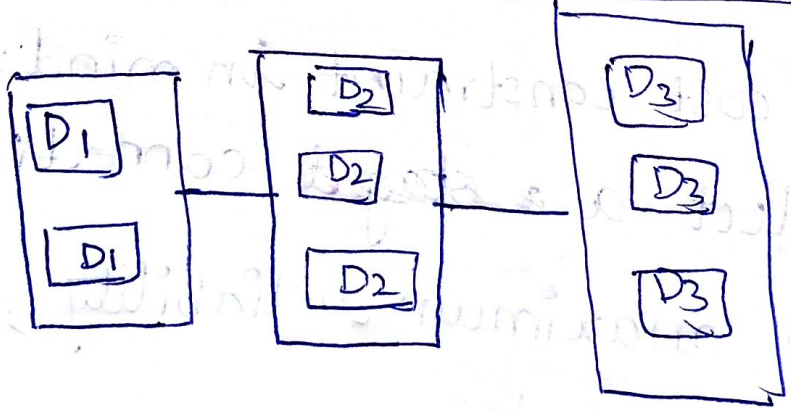
$$u_2 = \left[(105 + 15 - 65) / 15 \right] = (3).67$$

→ 3 copies of D_2

$$u_3 = \left[(105 + 20 - 65) / 20 \right] = 3$$

→ 3 copies of D_3

* reliability of m copies at a certain stage $= \left(1 - (1 - r_i)^m\right)$



* Principle of dominance:

Cost ↑ses but reliability↓

$S_0^1 = \{(30, 0.9)\}$ → 1st stage ; 1 copy

copy

$S_2^1 = \{(60, (1-(1-0.9)^2)\} = \{(60, 0.99)\}$

$S^1 = \{S_1^1, S_2^1\}$ → complete stage 1

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

$S_1^2 = \{(15, 0.8)\}$

$S_2^2 = \{(30, (1-(1-0.8)^2)\} = \{(30, 0.96)\}$

$S_3^2 = \{45, (1-(1-0.8)^3)\} = \{(45, 0.992)\}$

* $S_1^2 + S^1 = \{(45, 0.72), (75, 0.792)\}$

$S_2^2 + S^1 = \{(60, 0.864), (90, 0.95)\}$

∴ cost exceeded

$S_3^2 + S^1 = \{(75, 0.89), (105, 0.98)\}$

∴ cost exceeded

∴ $c_1 < c_2$ acc. to principle of dominance

$S^2 = \{(45, 0.72), (60, 0.864), (75, 0.792), (75, 0.89)\}$.

$S_1^3 = \{(20, 0.5)\}$

$S_2^3 = \{(40, (1-(1-0.5)^2)\}$

$= \{(40(1-0.25))\} = \{(40, 0.75)$

$S_3^3 = \{(60, (1-(1-0.5)^3)\}$

$= \{(60, 0.875\}$

0.125
1.000
0.125
0.875

$S_1^3 + S^2 = \{(65, 0.36), (80, 0.432),$
$(95, 0.445)\}$

$S_2^3 + S^2 = \{(85, 0.54), (100, 0.648), (115, 0.667$

cost exceeded

$S_3^3 + S^2 = \{(105, 0.63), (120, 0.756), (135, 0.7$

cost exceeded.

$S^3 = \{(65, 0.36), (80, 0.432), (85, 0.545),$
$(95, 0.445), (100, 0.648),$
$(105, 0.63)\}.$

(Law of dominance)

*Max. reliability = $0.648 \in S_2^3$.

Cost = 100

Stage-3 : 2 copies $\cong$ 40

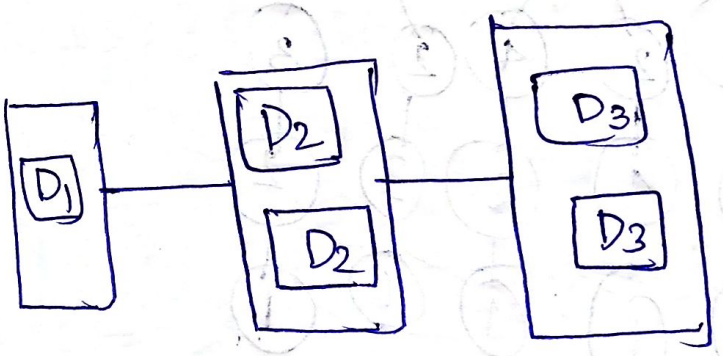
100 - 40 = (60, —)

Stage-2 : $60 \in S_2^2$: 2 copies $\cong$ (30, —)

60 - 30 = 30.

Stage-1 : $30 \in S_1^1$: 1 copy $\cong$ (30, —)

*Design



→ Maximize: $\prod_{1 \le i \le n} \phi_i(m_i)$

Subjected to $\sum_{1 \le i \le n} c_i m_i \le c$

where $m_i \ge 1 \; ; 1 \le i \le n$.

24/05/23

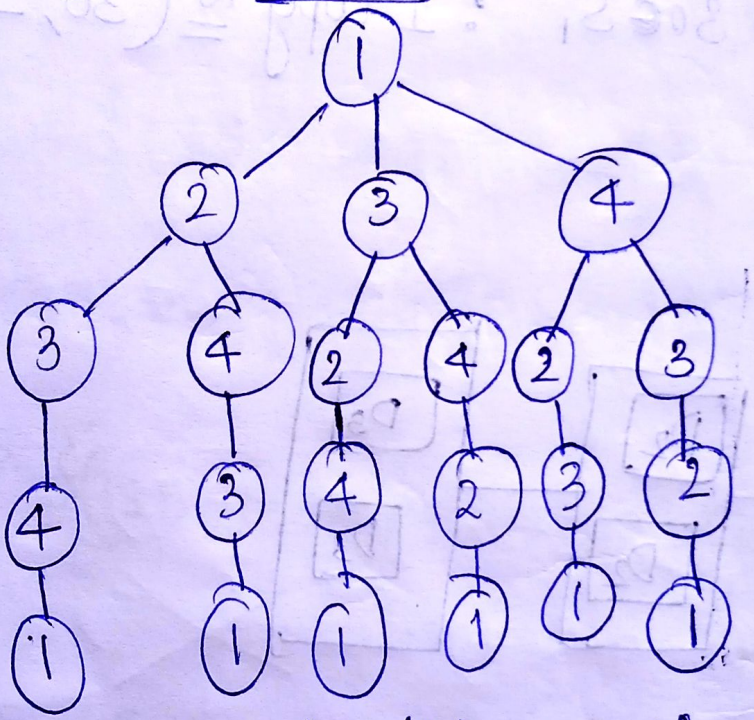
Travelling Sales person problem:

$G = (V, E)$ directed graph

Source to source again.

* This problem deals with tour travelled with minimum cost.

→ Brut Force Method:

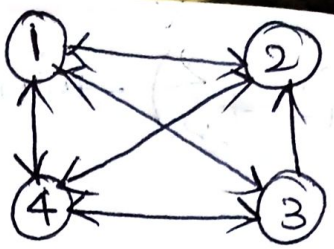


→ This is not a feasible solution:

By dynamic programming; using the cost function:

$$g(i, s) = \min_{k \in S} \{c(i, k) + g(k, S - \{k\})\}$$

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



$g(2,\phi) = c(2,1) = 5$

$g(3,\phi) = c(3,1) = 6$

$g(4,\phi) = c(4,1) = 8$

$g(2,\{3\}) = c(2,k) + g(3,\phi)$

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

$g(2,\{4\}) = c(2,4) + g(4,\phi)$

$= 10 + 8 = \underline{18}$

$g(3,\{2\}) = c(3,2) + g(2,\phi)$

$= 13 + 5 = \underline{18}$

$g(3,\{4\}) = c(3,4) + g(4,\phi)$

$= 12 + 8 = \underline{20}$

$g(4,\{2\}) = c(4,2) + g(2,\phi)$

$= 8 + 5 = \underline{13}$

$g(4,\{3\}) = c(4,3) + g(3,\phi)$

$$g(2,\{3,4\}) = \min\{\, c(2,3) + g(3,4),\ c(2,4) + g(4,3)\};$$

$$= \min\{29, 25\}$$

$$= 25 \quad (k=4)$$

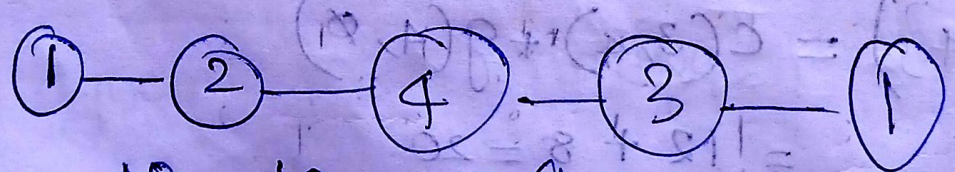
$$g(3,\{2,4\}) = 25 \quad (k=4)$$

$$g(4,\{2,3\}) = 23 \quad (k=2)$$

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

$$= \min\{35, 40, 43\}$$

$$= 35 \quad (k=2)$$



minimum cost = 81.

* Longest Common Subsequence:

$S_1 = abcfde$

$S_2 = bdef$

* → Finding the string which has longest length which is common to both.

→ This checking for common sequence can be done only in one direction.

Ex: S_1 = stone.

S_2 = Longest

		0	1 (l)	2 (o)	3 (n)	4 (g)	5 (e)	6 (s)	7 (t)
	0	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	0	1	1
t	2	0	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	1	2
n	4	0	0	1	2	2	2	2	2
e	5	0	0	1	2	2	3	3	3

Subsequence = one

* if (A[i] == B[j])

$LCS[i,j] = 1 + LCS[i-1, j-1]$

else

$LCS[i,j] = \max(LCS[i-1,j], LCS[i,j-1])$

$L[u] = \min\{df[u], \min\{L[w] \mid w \text{ is a child of } u\}, \min\{df[w] \mid (u,w) \text{ is a back edge}\}\}$

29/05/2023

UNIT-4

BACK-TRACKING:

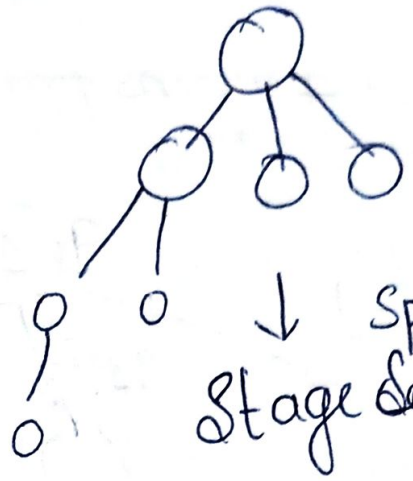
4 queens: 4 × 4 chessboard is given; placing each queen in each row; but any 2 queens should not be in same column (or) diagonal.

→ Back tracking deals with finding all the solutions for a certain problem.

→ Brute Force: Deals with finding sol? with the help of bounding fn. and check for all the possibilities.

*Implicit constraints.

*Explicit Constraints.



↓ Space.

Stage Tree

*Algorithm Backtrack(k)

{ for (each x[k] ∈ T(x[1], ..., x[k-1])) do

{ if (Bk(x[1], x[2], ..., x[k]) ≠ 0)

then

{ if (x[1], x[2], ... x[k] is a path to an answer node).

then write(x[1:k]);

if (k < n) then Backtrack(k+1);}}

}

*N Queens problem:

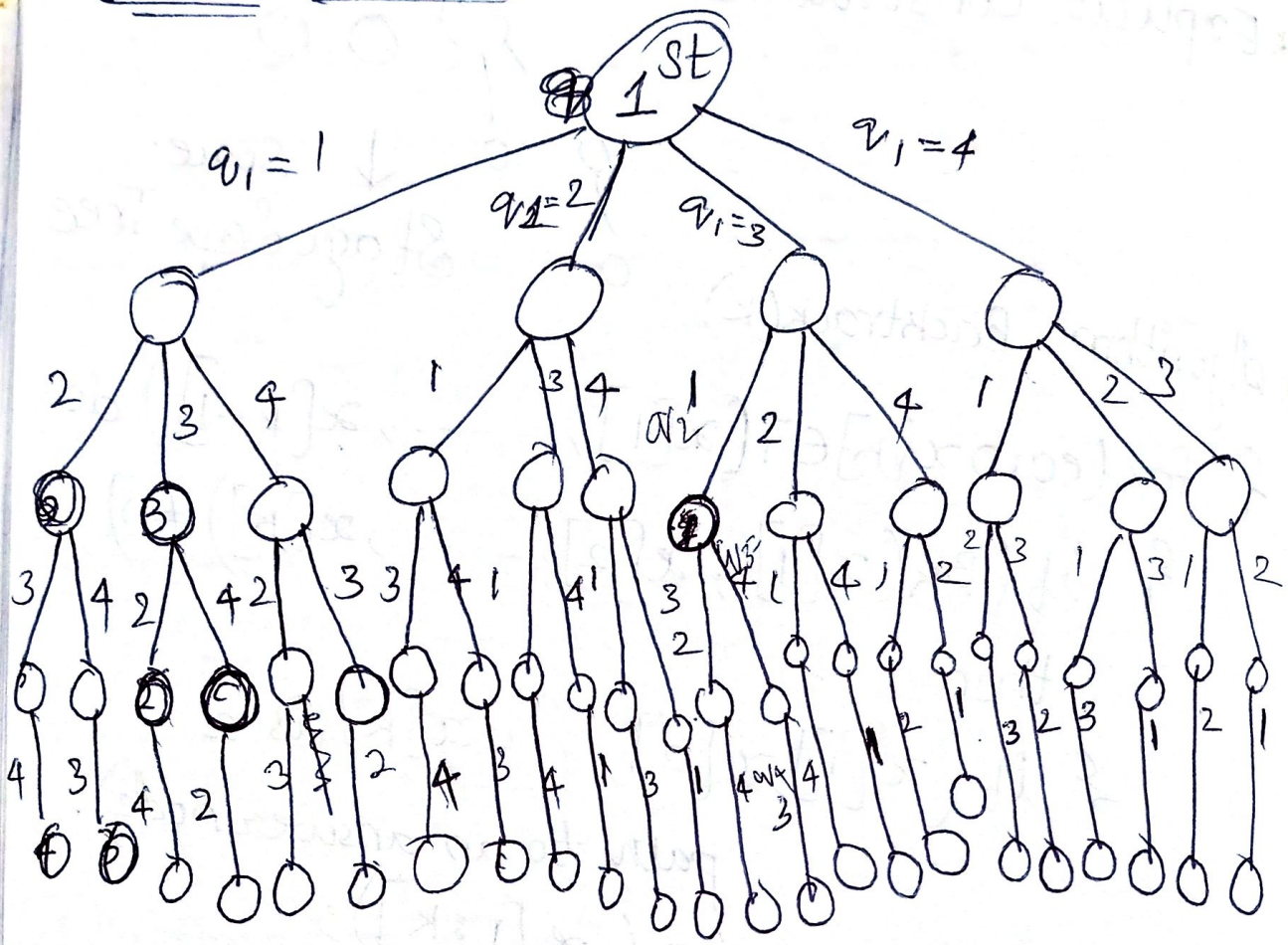
N×N → chess board is given

N → queens are given

Bounding function:

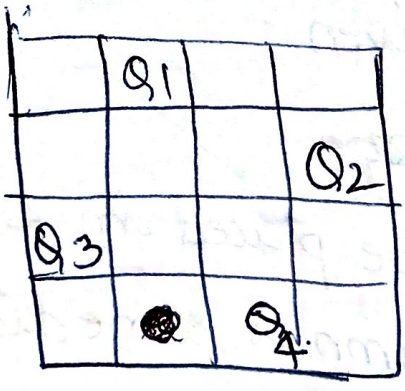
No 2 queens can be placed in same row, same column or same diagonal.

*Consider 4 queens problem.

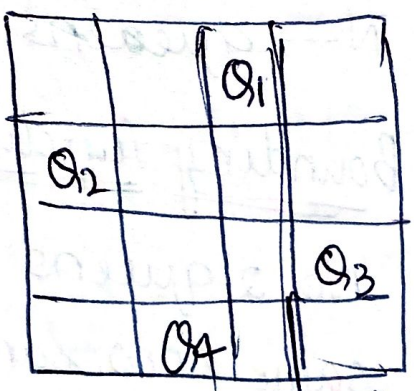


* $1 + \sum_{i=0}^{3} \left(\prod_{j=0}^{i} (4-i) \right)$

$$1 + \sum_{i=0}^{N} \left(\prod_{j=0}^{n-i} (N-i) \right)$$



2, 4, 1, 3



(3, 1, 4, 2).

→ Algorithm NQueens(k, n)

```
{ for i := 1 to n do
  { if Place(k, i) then
    { x[k] := i;
      if (k = n) then write (x[1:n]);
      else
        NQueens(k+1, n); }}}.
```

→ Algorithm Place(k, i)

```
{ for j := 1 to k-1 do
  { if ((x[j] = i) or (Abs[x[j] - i)
                       = Abs(j - k)))
      then return false;
    return true; }
```

31/05/2023

→ Graph Colouring problem:

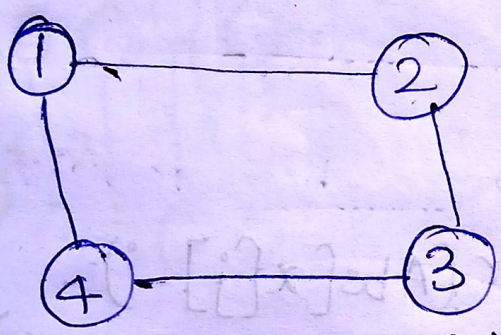
- → Decision problem (Yes/No kind)
- → Optimization problem (finding optimal soln).

$G = (V, E)$, given a set of colours.
→ Bounding function

* We must colour the graph such that no 2 adjacent vertices have same colour.

→ We must check for minimum no. of colours to colour the graph
↓
(called chromatic no. for graph)

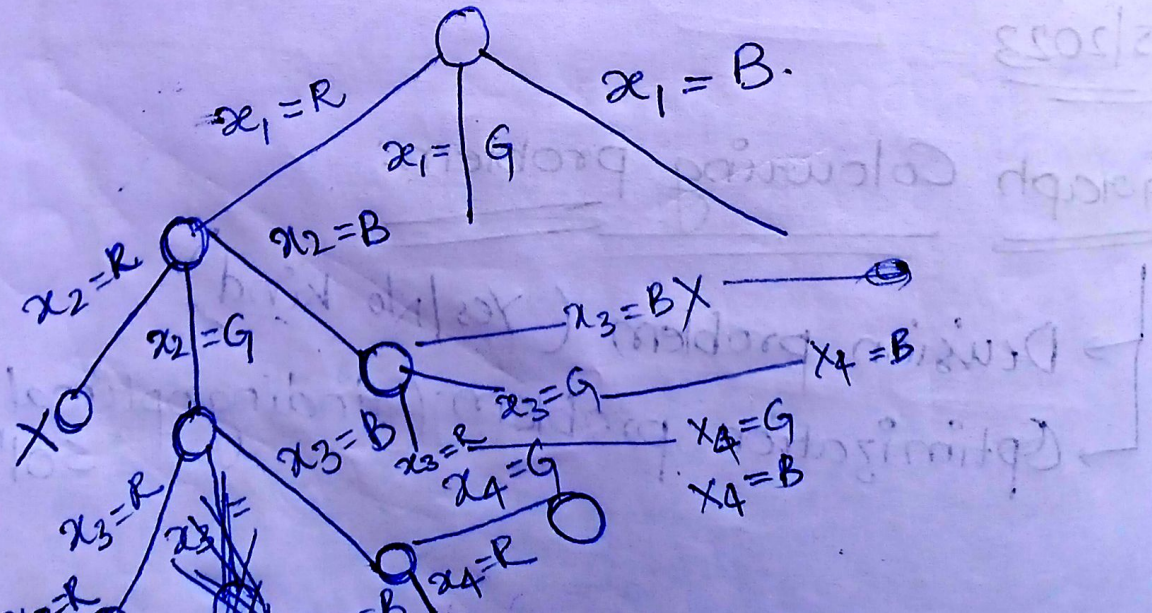
* Example:



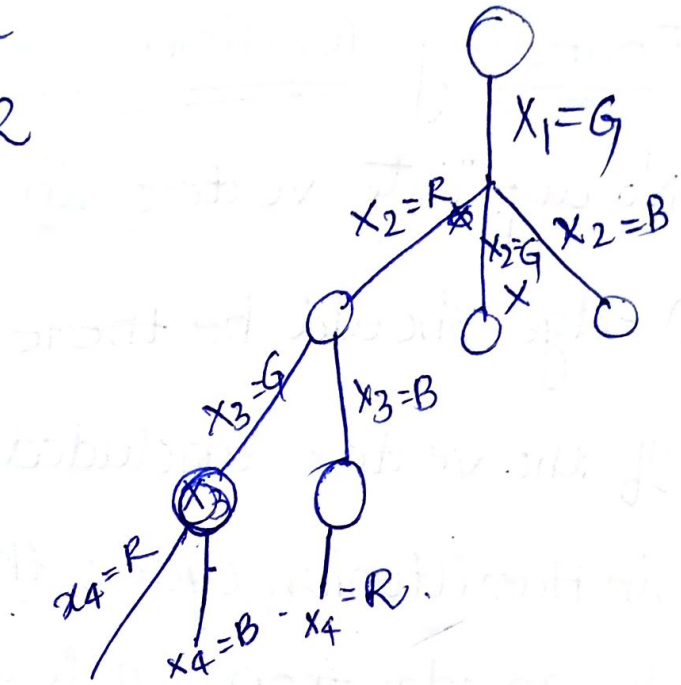
$n = 4$ Colours $= \{R, G, B\}$

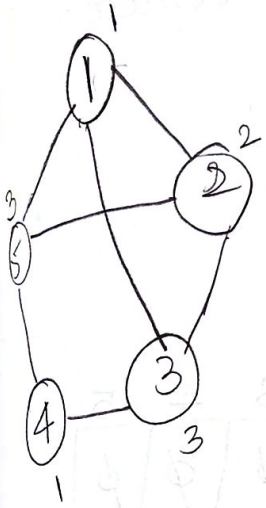
If $C \rightarrow$ chromatic no. of graph
$n \rightarrow$ no. of vertices.

$\therefore$ time complexity $\propto C^n$



→ RGRG | GRGR
RGRB | GRBR
→ RGBG
→ RBRG.
→ RB.RB
→ RBGB





→ Chromatic no = 3

∴ In any graph, the min no. of colours required from the set of colours to colour all the nodes checking with bounding function helps in get chromatic no.

*Applications:

→ Map Colouring.

→ Register allocation & assignment.

09/06/2023

→ Hamiltonian Cycle:

G = (V, E)

V_1, V_2, V_3, V_4, V_5.

* Bounding function

1) No duplicate vertex in Hamiltonion cycle.

2) edge should be there from k to (k+1)

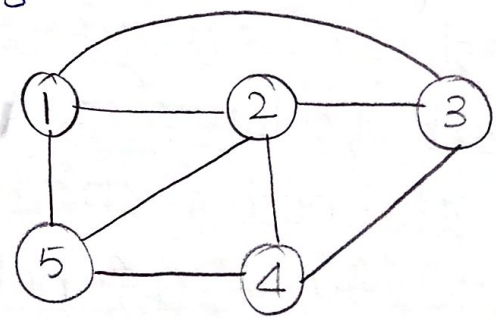
3) If the vertex included is n^{th} vertex in Hamiltonion cycle; then there should be an edge from n to ~~to~~ the 1^{st} vertex.

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

1 ✗2

→ {1, 2} ---

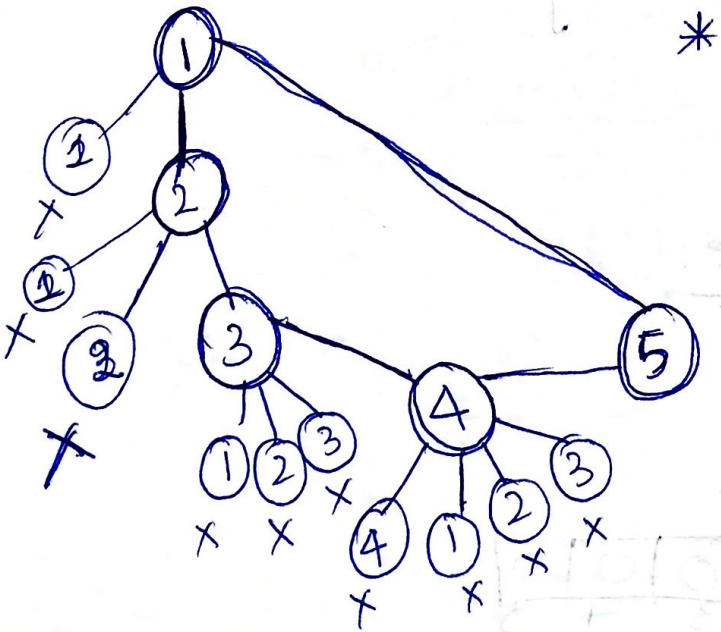
*



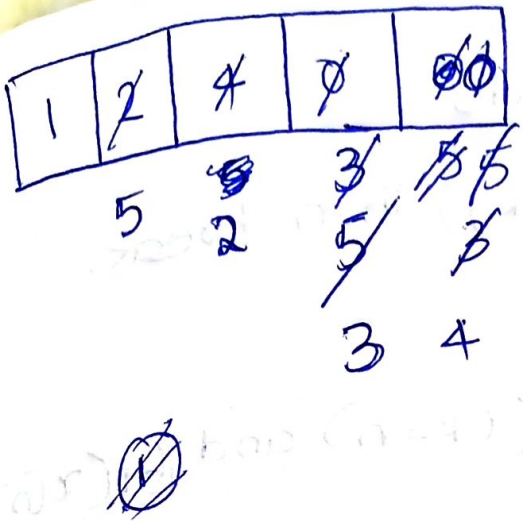
1	2	3	4	5
~~0~~	~~0~~	~~0~~	~~0~~	~~0~~

1 ✗ ✗ ✗ ✗
2 ✗ ✗ ✗
3 ✗ ✗
4 ✗
5

Set = {1, 2, 3, 4, 5, 1}



* After finding a sol^n we must check for alternate in each step.



ans = {1, 5, 2, 4, 3}

= {1 2 5 3 4 3 1}

= {1 3 4 5 2 1}

* Algorithm:

```
Algorithm Ham(k)
{ repeat {
    NextValue(k);
    if (x[k] = 0) then
        return;
    if (k = n) then
    write(x[1:n])

    else
        Ham(k+1) } until(false); }
```

```
Algorithm nextvalue(k)
{ repeat {
  x[k] := (x[k]+1) mod (n+1)

  if(x[k]=0) then return;
if (G[x[k-1], x[k] ≠ 0) then
```

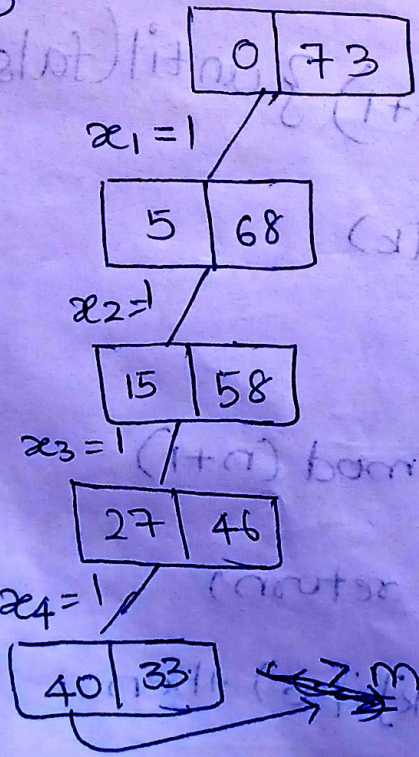
```
for j:=1 to k-1 do
    if(x[j]=x[k]) then break.
if(j=k) then
    if((k<n) or((k=n) and Bn(x[n],
                                x[i]≠0))
        then return;
    }
} until(false)
}
```

* Sum of subsets:

- fixed size soln = no. of elements given
- variable size soln = no. of elements selected

w[1-6] = {5, 10, 12, 13, 15, 18} = 73

m = 30.



*Bounding function:

$$\sum_{i=1}^{k} (w_i x_i + w_{k+1} x_{k+1}) \leq m$$

$$\sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} w_i > m.$$

BRANCH & BOUND:

→ Minimization of bounding function.

→ Follows BFS

Types of branch & bound: — (DSP, 0/1 Knapsack)

1) FIFO BB (queue)
2) LIFO BB (stack)
3) LC BB (Least cost)

*Control Abstraction:

```
Listnode = record {
        Listnode * next, *parent;
        float cost; }

Algorithm LCsearch( )
{ if *t is an answer node then
    output *t and return;
  E:=t;
  Initialize list of live nodes to
  be empty repeat {
```

```
{ for each child x of E do
  { if x is an answer node then
      output the path from x to t
      and return;
    add (x);
    x ← parent = E }
  If there are no more live nodes
  then
  { write ('No answer node');
      return;
    }
  E := least ( ) (least value from list
                  of values)
  }
until (false);
}
```

Travelling Sales Person

* TSP with least Cost Branch & Bound:

15/06/2023

	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	4	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞

⇒ Reduced cost matrix

1 row & 1 col is 0

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 15 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

$R_1 = 10$
$R_2 = 2$
$R_3 = 2$
$R_4 = 3$
$R_5 = 4$

reduce
$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 12 & \infty & 14 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 15 & \infty & 0 \\ 11 & 0 & 3 & 12 & \infty \end{bmatrix}$$

(i,j) edge ith row & jth col to ∞ and (j, 1) too

to go from 1 to 2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ \infty & \infty & \infty & 0 & 2 \\ 0 & \infty & 12 & \infty & 0 \\ 5 & \infty & 0 & 12 & \infty \\ 11 & \infty & & & \end{bmatrix}$$

All col & row has 0 so no need to reduce: $C(2) = C(1) + A[i,j] + x$

$= 25 + 10 + 0$ (no reduction)

$= 35$

to go from 1 to 3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix}$$

$C_1 - 11 \Rightarrow$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

$C(3) = C(1) + A[i,j] + r$

$= 25 + 17 + 11 = 53.$

to go from 1 to 4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

$C(4) = C(1) + A[i,j] + r$

$= 25 + 0 + 0 = 25.$

to go from 1 to 5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

$R_2 \rightarrow R_2 - 1$; $R_4 \rightarrow R_4 - 3$

$C(5) = 25 + 1 + 5 = 31$

We keep i^{th} row & j^{th} col to ∞ because we dont want to change their values as we are moving across them. After every movement we try to reach easily.