__/__/____

ADA

Greedy Algorithm

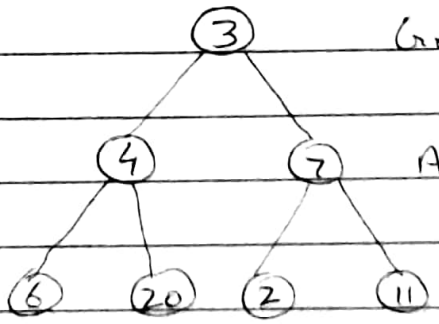
A greedy algo is a Strategy that makes the optimal choice at each stage with hope of finding a global optimal.

Pros: Simple, easy to implement, run fast.
Cons: do not always yield optimal Solution.

Characteristics of greedy algorithm

1) used to solve optimization problem.
2) Most general, straight forward method to solve a problem.
3) Always makes the choice that looks best at the moment i.e it makes a "locally optimal choice in the hope that this choice will lead to a overall globally optimal Solution".
4) It does not always yields an optimal Solution but for many problem they do.
5) Once any choice of input from C is rejected then it never considered again.

Eg:



Minimization

Greedy:
3 + 4 + 6 = 13

Actual:
3 + 7 + 2 = 12

Maximization:

Greedy: 3 + 7 + 11 = 21
Actual: 3 + 4 + 20 = 27

Time complexity = $n \log n + n = (n \log n)$

__/__/____

→ Activity Selection Problem
→ Huffman Coding
→ Knapsack Problem
→ finding Minimum Spanning tree.
  ↳ Krushkal's algo
  ↳ Prim's algo
→ Single Source Shortest path
  ↳ Dijkstra's algo
  ↳ Bellmenford algo

⇒ Activity Selection Problem

There are n different activity are given (with their Starting and ending time. Select maximum number of activity to solve by a single person.

1) Sort the activity with their ending time.
2) Find compatible activity and add to list.

Example:

A_i	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
S_i	1	2	4	1	5	8	9	11	13
T_i	3	5	7	8	9	10	11	14	16

$n = 9$

Greedy approach : $A = \{A_1, A_3, A_6, A_8\}$

General approach : $A = \{A_2, A_5, A_7, A_8\}$

Total 4 activity.

__/__/__

Algorithm:

Greedy-activity (S, f)

1) $n \leftarrow$ Length (S)
2) $A \leftarrow \{1\}$
3) $J \leftarrow 1$
4) for $i \leftarrow 2$ to n
5) do if $S_i \geq f_J$
6) then $A \leftarrow A \cup \{i\}$
7) $J \leftarrow i$
8) Return A

Q Given 10 activity along with their start & finish time.

$S = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}\}$

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
S_i =	1	2	3	4	7	8	9	~~10~~ 9	11	12
f_i =	3	5	4	7	10	9	11	13	12	14

loc:	1	2	3	4	5	6	7	8	9	10
S_i	1	3	2	4	8	7	9	11	9	12
f_i	3	4	5	7	9	10	11	12	13	14
	A_1	A_3	A_2	A_4	A_6	A_5	A_7	A_9	A_8	A_{10}

$n = 10$

__/__/____

Huffman Coding

Huffman Coding used to compress the data upto 90%. Suppose we've 100 character data having six different character.

	A	B	C	D	E	F	
Freq	45	13	12	16	9	5	=100

ASCII 000100.....001001010..... = 100 x 8 bit = 800 bit

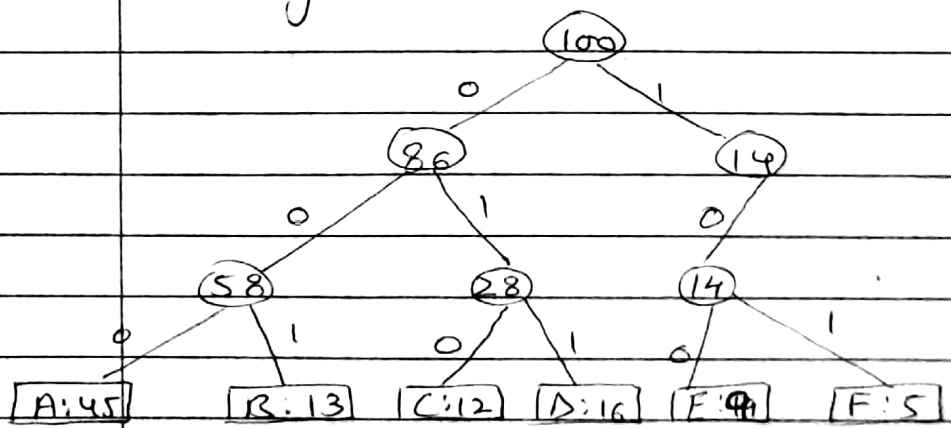
fix length = 000 001 010 011 100 101 = 100 x 3 = 300 bit

van length = 0 101 100 111 1101 1100

= 45x1 + 13x3 + 12x3 + 16x3 + 9x4 + 5x4

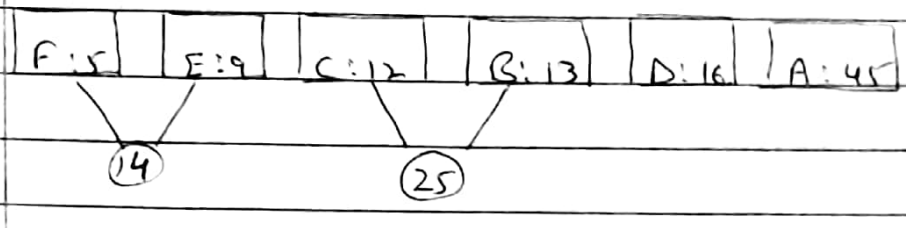
= 224 bit

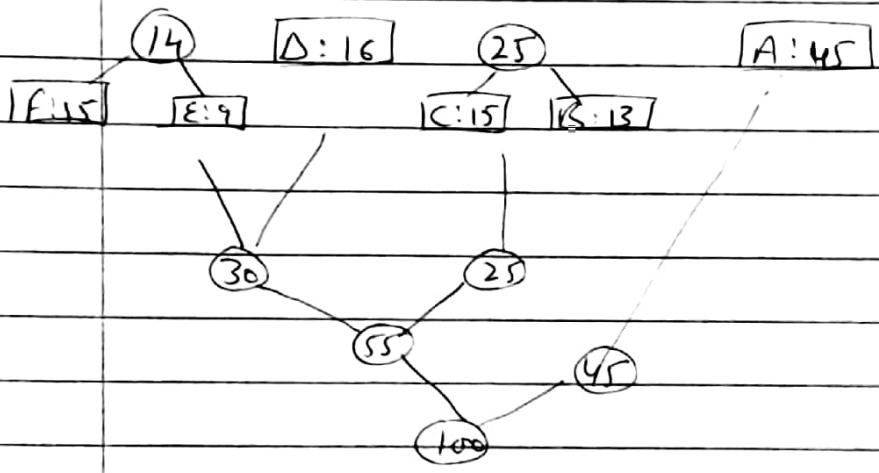
Fixed length

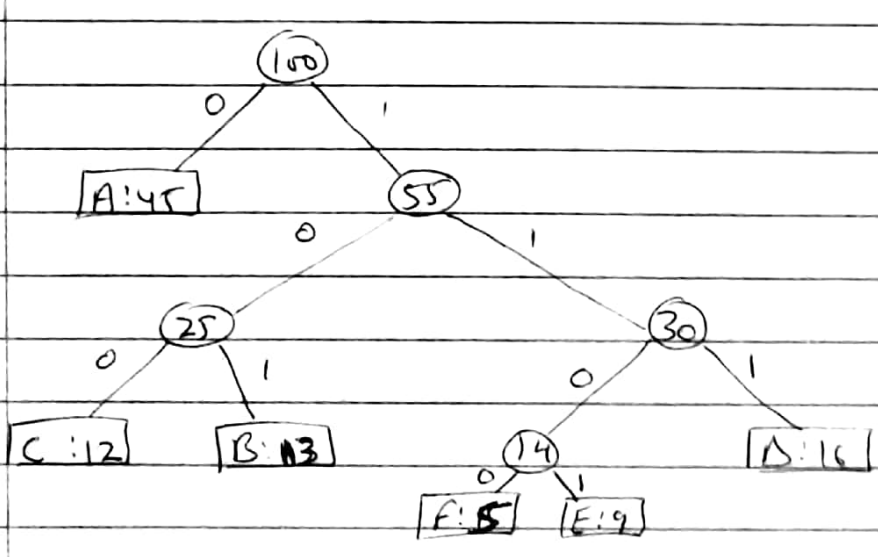


= 300 + 6x8 + 18

= 300 + 48 + 18 = 366 bit







VL

= 224 bit

= 224 + 48 + 18 = 290 bit

Note: Huffman coding use variable length.

__/__/____

Knapsack Problem (Fraction)

We have n object and a knapsack (bag). Each object have some profit and weight. The capacity of knapsack (bag) is m.

Problem: Fill the bag with object that maximize the profit.

Example: $n = 3 \qquad m = 20$

$(P_1, P_2, P_3) = (25, 24, 15)$

$(w_1, w_2, w_3) = (18, 15, 10)$

Solution:

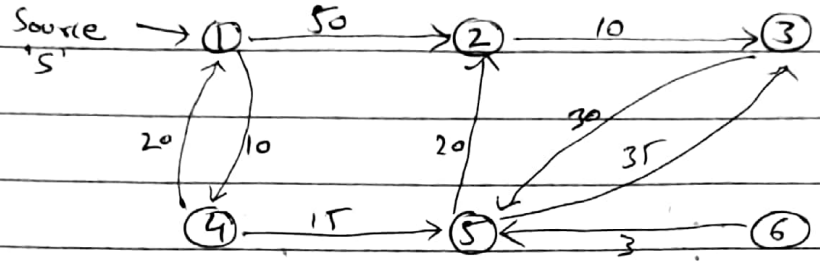
	x_1	x_2	x_3
Object	1	2	3
Profit	25	24	15
Weight	18	15	10
Profit/weight	1.38	1.6	1.5

	x_1	x_2	x_3	$\sum w_i x_i$	$\sum p_i x_i$
Max Profit:	1	$\frac{2}{15}$	0	$18 + \frac{2}{15} \times 15 = 20$	$25 \times 1 + 24 \times \frac{2}{15} = 28.2$
Min weight	0	$\frac{10}{15}$	1	$10 \times 1 + \frac{10}{15} \times 15 = 10 + 10 = 20$	$15 \times 1 + 24 \times \frac{10}{15} = 31$
Profit/weight (Max)	0	1	$\frac{5}{10}$	$15 \times 1 + \frac{5}{10} \times 10 = 20$	$24 \times 1 + \frac{5}{10} \times 15 = 31.5$

$\Rightarrow$ Profit/weight gives optimal solution.

Single Source Shortest Path

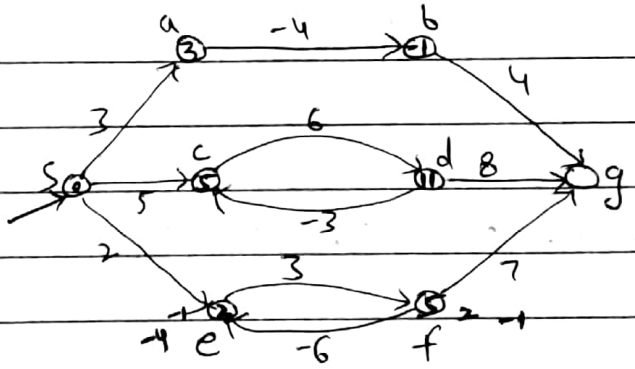
Given a graph $G = (V, E)$ we want to find a shortest path from a given source vertex $s \in V$ to each vertex $v \in V$

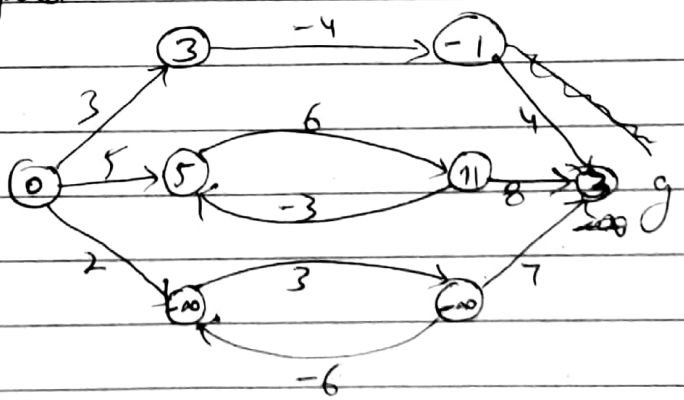


Path	Length
(1,4) : 1,4	10
(1,2) : 1,4,5,2	45
(1,3) : 1,4,5,2,3	55
(1,5) : 1,4,5	25
(1,6)	∞

Terminology

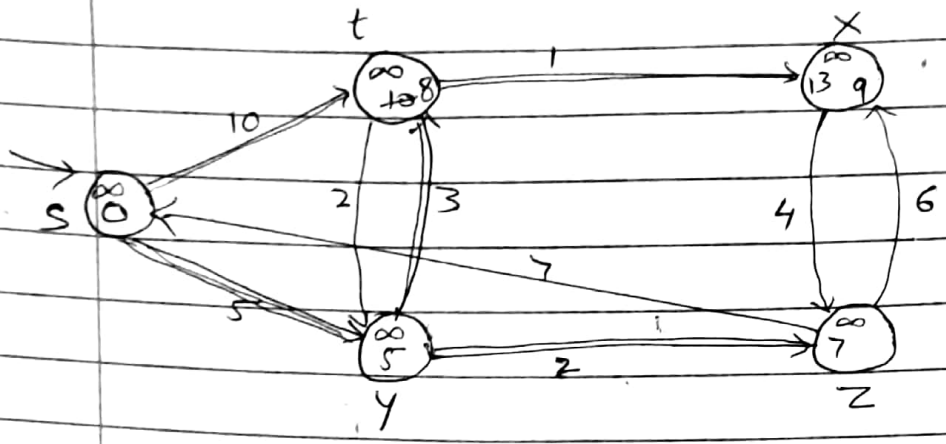
1) vertex
2) Edge
3) weighted edge
   1) Positive
   2) Negative
4) Cycle
   1) Positive weighted
   2) Negative weighted





1) Dijkstras Algorithm (Negative weight but not negative cycle)
2) Bellman ford Algorithm (Positive weight)

DIJKSTRAS Algorithm
for the problem in which all egdes weight are non negative.



DIJKSTRA (G, w, s)
1) INITIALISE - SINGLE - SOURCE (G, S)
2) $S \leftarrow \phi$
3) $\phi \leftarrow V[G]$
4) while $Q = \phi$
   do $u \leftarrow$ Extract min (Q)
      $S \leftarrow S \cup \{u\}$
      for each vertex $v \in Adj(u)$
         do RELAX (u, v, w)

INITIALISE - SINGLE - SOURCE (G, S)
for each vertex $v \in V[G]$
   do $d[v] \leftarrow \infty$
      $\pi[v] \leftarrow$ NIL

RELAX (u, v, w)

if d[v] > d[u] + w(u,v)

then d[v] ← d[u] + w(u,v)

π[v] ← u

S = [S, y, z, t, x]

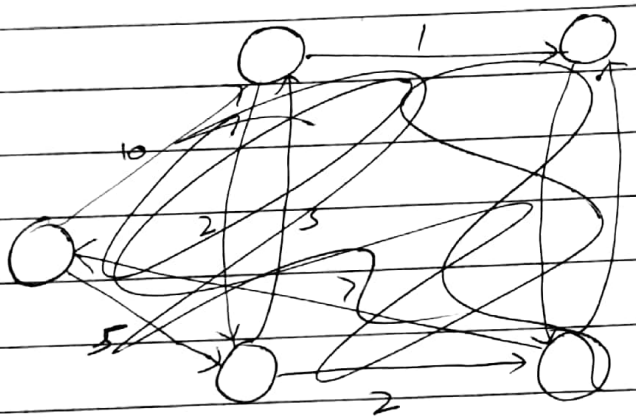
Q = [s, t, x, y, z]

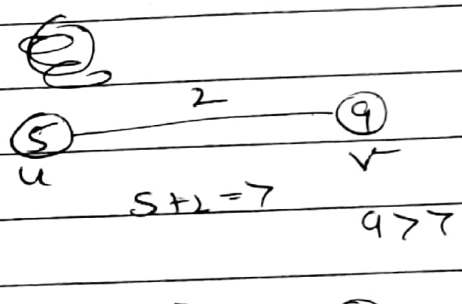
Q = [t, x, y, z]

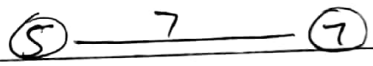
Q = [t, x, z]

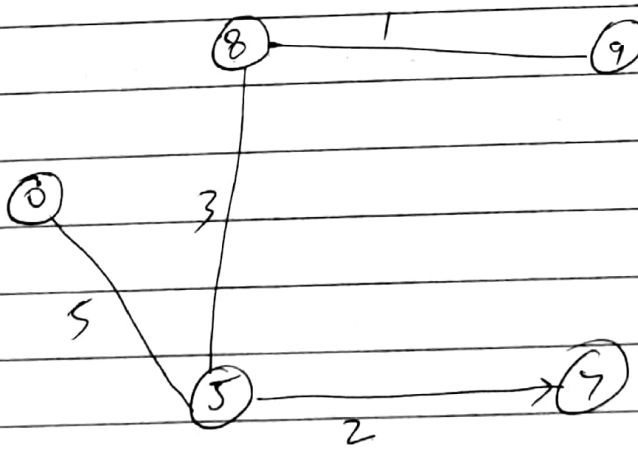
Q = [t, x]

Q = [x]



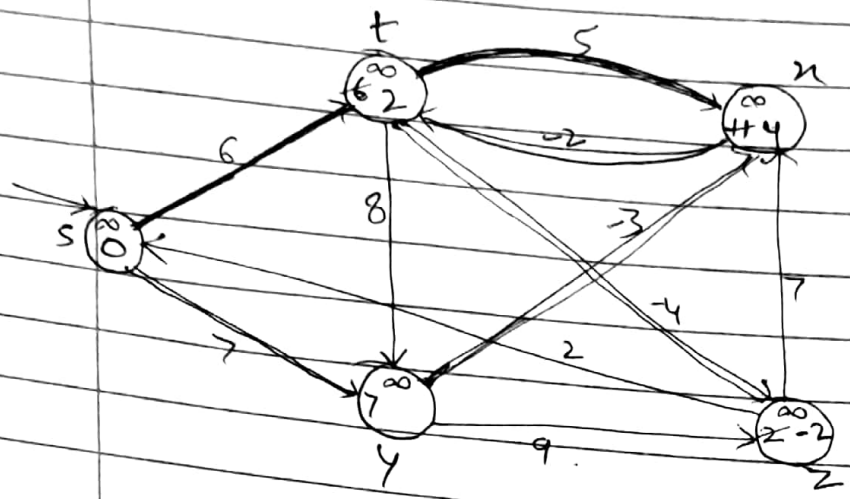






Bellman-ford Algorithm

Solve the problem with weight may be negative but not negative cycle.



BELLMAN-FORD (G, w, s)

1) INITIALIZE - SINGLE - SOURCE (G, s)
2) for i ← 1 to |V[G]| - 1
3) do for each edge (u, v) ∈ E[G]
4) do RELAX (u, v, w)
5) for each edge (u, v) ∈ E[G]
6) do if d[v] > d[u] + w[u, v]
7) then Return FALSE
8) RETURN TRUE

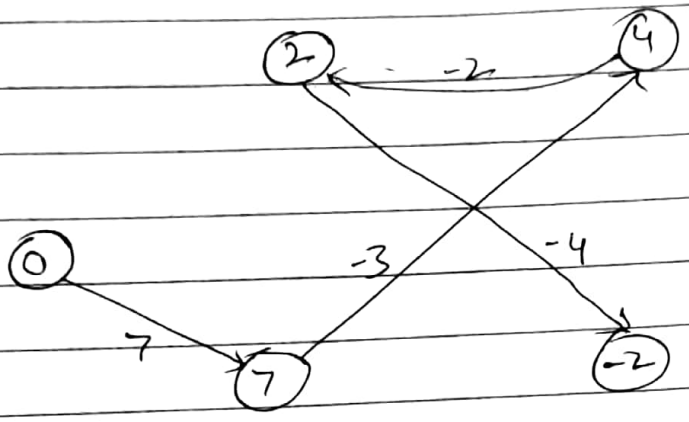
-ve weight cycle

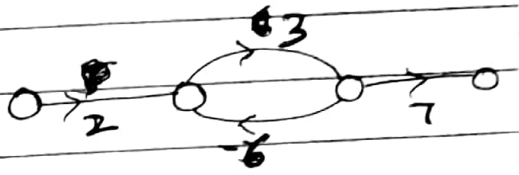
INITIALIZE - SINGLE - SOURCE (G, s)

1) for each vertex v ∈ V[G]
2) do d[v] ← ∞
3) π[v] ← NIL
4) d[s] ← 0

RELAX (u, v, w)

1) if d[v] > d[u] + w(u, v)
2) then d[v] ← d[u] + w(u, v)
3) π[v] ← u





Dynamic Programming

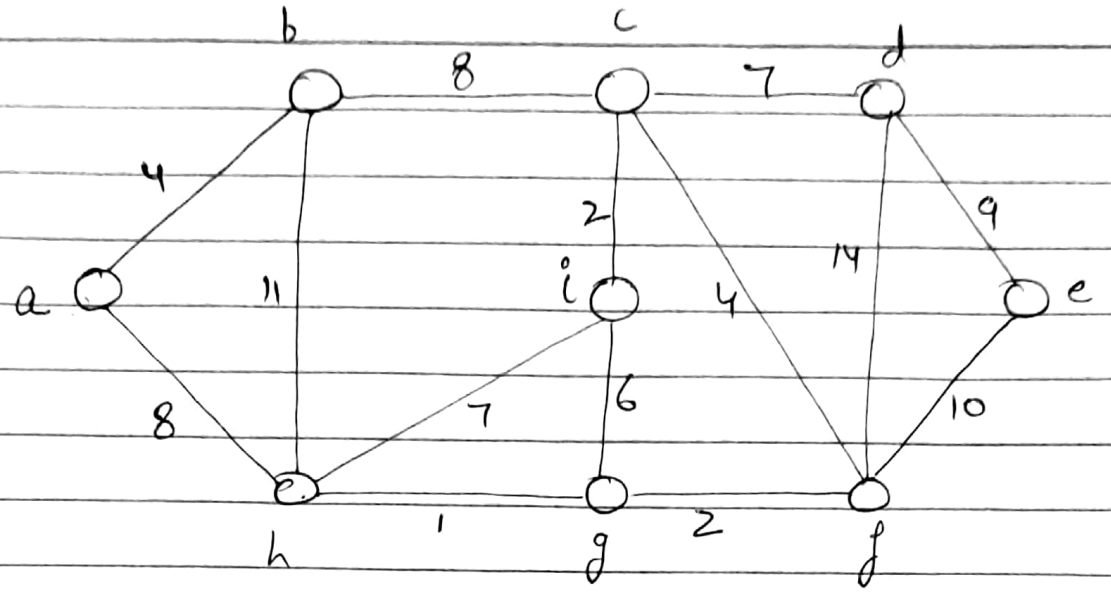
→ It is a technique for solving recursively or non recursively by memorization or tabulation method.

→ Recomputation is avoided, Reusability is involved.

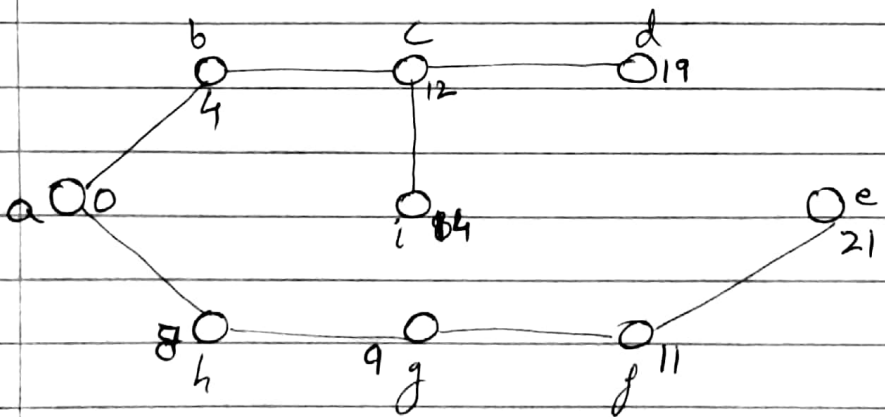
→ Greedy approach does not give optimal solution guarantee but Dynamic Programing gives guarantee of optimal solution by using principle of optimality.

Principle of optimality: A problem is said to satisfy principle of optimality if the sub solution of an optimal solution of the problem are themselves optimal solution for their subproblems.

__/__/____

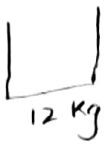


a	b	c	d	e	f	g	h	i
~~∞~~	∞	∞	∞	∞	∞	∞	∞	∞
[0]	∞	∞	∞	∞	∞	∞	∞	∞
	[4]$_a$	∞	∞	∞	∞	∞	8_a	∞
		12_b	∞	∞	∞	∞	[8]$_a$	∞
		12_i	∞	∞	∞	[9]$_h$	~~∞~~	15_h
		12_b	∞	∞	[11]$_g$			15_h
		[12]$_b$	25_f	21_f				15_h
			19_b	21_f				[14]$_b$
			[19]$_b$	21_f				
				[21]$_f$				



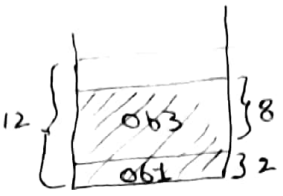
2^n → no of Possibilities

0/1 Knapsack Problem (0 - absent, 1 - present)



Object	ob1	ob2	ob3
weight	2	4	8
Profit	20	25	60
P/w	10	6.2	7.5

Knapsack Capacity (M) = 12



1) Brute force method:

0 0 0 → NP
0 0 1 → 60
0 1 0 → 25
0 1 1 → 85
1 0 0 → 20
1 0 1 → 80
1 1 0 → 45
1 1 1 → NP

$P = \{1, 2, 5, 6\}$ $M = 8$
$n = 4$
$w = \{2, 3, 4, 5\}$

2) By Dynamic Programing:

→ capacity of bag

P_i	w_i	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3
5	4	3	0	0	1	2	5	5	6	7
6	5	4	0	0	1	2	5	6	6	7	8

for last row

$$v[i, w] = \max\{v[i-1, w], v[i-1, w - w[i]] + P[i]\}$$

$$v[4, 1] = \max\{v[3, 1], v[3, 1-5] + 6\}$$
$$3, -4$$

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

$$v[4,5] = \max\{v[3,5], v[3, 5-5] + 6\}$$
$$3, 0$$
$$5, 0 + 6$$
$$5, 6$$

$$v[4,6] = \max\{v[3,6], v[3, 6-5] + 6\}$$
$$3, 1$$
$$6, 0 + 6$$

$$v[4,7] = \max\{v[3,7], v[3, 7-5] + 6\}$$
$$= 7, 1 + 6$$

$$v[4,8] = \max\{v[3,8], v[3, 8-5] + 6\}$$
$$7, \quad 3, 3$$
$$2 + 6$$

Dynamic Programming

→ It is a technique for solving recursively or non recursively by memorization or tabulation method.

→ Recomputation is avoided, Reusability is involved.

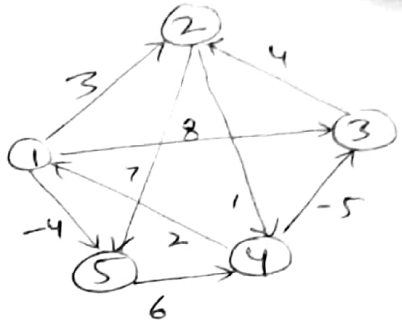
→ Greedy approach does not give optimal solution guarantee but Dynamic Programing gives guarantee of optimal solution by using principle of optimality.

Principle of optimality: A problem is said to satisfy principle of optimality if the sub solution of an optimal solution of the problem are themselves optimal solution for their subproblems.

All pair Shortest Path - Floyd warshall algorithm

$D_0 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0



Floyd warshal (w) — $O(V^3)$

1) n ← rows (w)
2) $D_0 \leftarrow w$
3) for k → 1 to n — (n)
4) do for i ← 1 to n — row — (n^2)
5) do for j ← 1 to n — col — (n^3)
6) do $D_{ij}(k) \leftarrow \min(d_{ij}(k-1), d_{ik}(k-1) + d_{kj}^{(k-1)})$
7) return D_n

$D_1 =$

	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞				0

$D'_{2,3} \leftarrow \min(d^0_{2,3}, d^0_{2,1} + d^0_{1,3})$
$\leftarrow \min(\infty, \infty + 8)$
$D'_{2,3} \leftarrow \infty$

$D'_{2,4} \leftarrow \min(d^0_{2,4}, d^0_{2,1} + d^0_{1,4})$
$\leftarrow \min(1, \infty + \infty)$
$D'_{2,4} \leftarrow 1$

$D'_{3,2} \leftarrow \min(d^0_{3,2}, d^0_{31} + d^0_{12})$
$\leftarrow \min(4, \infty + \infty)$
$\leftarrow 4$

$D'_{4,2} \leftarrow \min(d^0_{4,2}, d^0_{41} + d^0_{12})$
$\leftarrow \min(\infty, 2 + 3)$
$\leftarrow 5$

$D'_{4,5} \leftarrow \min(d^0_{4,5}, d^0_{4,1} + d^0_{1,5})$
$\leftarrow \min(\infty, 2 + -4)$
$\leftarrow -2$

$D^2 =$

	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^3 =$

$D^2_{1,3} = D_{12,23}$ $\quad D_{3,4} = D_{32,24}$
$D_{1,4} = D_{12,24}$ $\quad D_{3,5} = D_{32,25}$
$D_{1,5} = D_{12,25}$ $\quad D_{4,1} = D_{42,21}$
$D_{3,1} = D_{32,21}$ $\quad P_{4,3} =$

$D^3 = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix}$ $D^4 = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & 5 & 1 & 6 & 0 \end{bmatrix}$ $D^5 = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix}$

Matrix chain multiplication

We are given a sequence of matrices. find the most effient way to multiply these matrices together.

$\langle A_1 \quad A_2 \quad A_3 \rangle$
$2\times3\quad 3\times2 \quad 2\times3$

$\langle A_1, A_2, A_3 \rangle = \langle A_1 \quad A_2 A_3 \rangle$

$\langle A_1 A_2 A_3 A_4 \rangle$

1) $(((A_1 A_2) A_3) A_4)$
2) $((A_1 A_2)(A_3 A_4))$
3) $((A_1 (A_2 A_3)) A_4)$
4) $(A_1 ((A_2 A_3) A_4))$
5) $(A_1 (A_2 (A_3 A_4)))$

$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}_{2\times2}$ $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}_{2\times2}$

$C = \begin{bmatrix} A_{11}B_{11} + A_{12} \times B_{21} & A_{11}B_{12} + A_{12} \times B_{22} \\ A_{21}B_{11} + B_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}$

Eg: $A_1 = 10 \times 100$
$A_2 = 100 \times 5$
$A_3 = 5 \times 50$

$((A_1 \times A_2) \times A_3) = ((10 \times 100 \times 5)_{10\times5} + (10 \times 5 \times 50)_{(A_3)_{5\times50}}$
$= 5000 + 2500$
$= 7500$

$(A_1 \times (A_2 \times A_3)) = (A_1)_{10\times100} + (100 \times 5 \times 50)$
$= (A_1)_{10\times100} + (25000)_{100\times50}$
$= (10 \times 100 \times 50) + 25000$
$= 50,000 + 25000$
$= 75000$

Matrix chain Multiplication

Consider following four matrices. Find optimal parenthesization of matrix chain multiplication.

matrix	order
A_1	20×30
A_2	30×50
A_3	50×10
A_4	10×5

$\langle A_1, A_2, A_3, A_4 \rangle$

$$\frac{1}{n} {}^{2(n-1)}C_{n-1}$$

$$= \frac{1}{4} \times {}^{2\times 3}C_3$$

$$= \frac{1}{4} \times {}^{6}C_3$$

$$= \frac{1}{4} \times \frac{6 \times 5 \times 4}{3!}$$

$$= 5$$

1) $((A_1 A_2) A_3) A_4)$
2) $((A_1 (A_2 A_3)) A_4)$
3) $(A_1 ((A_2 A_3) A_4))$
4) $((A_1 A_2) (A_3 A_4))$
5) $(A_1 (A_2 (A_3 A_4)))$

} optimal sol.

Ordered Matrix chain Multiplication

$P \langle 20, 30, 50, 10, 5 \rangle$

$P_0 \; P_1 \; P_2 \; P_3 \; P_4$

$A_1 \times A_2 \times A_3 \times A_4$

$20 \times 30 \quad 30 \times 50 \quad 50 \times 10 \quad 10 \times 5$

$P_0 \quad P_1 \quad P_2 \quad P_3 \quad P_4$

Cost Matrix M → J

	1	2	3	4
1	0	30,000	21,000	13,000
2		0	15,000	10,000
3			0	2500
4				0

Marker Split matrix S

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

$$m[i,j] = \min_{i \le k < j} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \}$$

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

$p < 20, 30, 50, 10, 5 >$

$p_0 \; p_1 \; p_2 \; p_3 \; p_4$

$$\Rightarrow m[1,2] = \min_{i \le k < j} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \}$$

$$m[1,2] = \min_{k=1} \{ m[1,1] + m[2,2] + p_0 p_1 p_2 \}$$

$$= \min \{ 0 + 0 + 20 \times 30 \times 50 \}$$

$$= \min \{ 30,000 \}$$

$$m[2,3] = \min \{ m[2,2] + m[3,3] + p_1 p_2 p_3 \}$$

$$0 \qquad 0 \qquad 30 \times 50 \times 10$$

$k = 2$ $= 15000$

$$m[3,4] = \min \{ m[3,3] + m[4,4] + p_2 p_3 p_4 \}$$

$$0 \qquad 0 \qquad 50 \times 10 \times 5$$

$k = 3$

$= 2500$

$m[1,3] = \min \begin{cases} m[1,1] + m[2,3] + P_0 P_1 P_3 & k=1 \\ m[1,2] + m[3,3] + P_0 P_2 P_3 & k=2 \end{cases}$

$= \min \begin{cases} 0 + 15000 + 20 \times 30 \times 10 \\ 30,000 + 0 + 20 \times 50 \times 10 \end{cases}$

$= \min \begin{cases} 15000 + 6000 \\ 30,000 + 10,000 \end{cases}$

$= \min \begin{cases} 21,000 \\ 40,000 \end{cases} = 21,000$

$m[2,4] = \min \begin{cases} m[2,2] + m[3,4] + P_1 P_2 P_4 & k=2 \\ m[2,3] + m[4,4] + P_1 P_3 P_4 & k=3 \end{cases}$

$= \min \begin{cases} 0 + 2500 + 30 \times 50 \times 5 \\ 15000 + 0 + 30 \times 10 \times 5 \end{cases}$

$= \min \begin{cases} 2500 + 7500 \\ 15000 + 1500 \end{cases}$

$= \min \begin{cases} 10,000 \\ 16,500 \end{cases} = 10,000$

$$m[1,4] = \min \begin{cases} m[1,1] + m[2,4] + P_0 P_1 P_4 & k=1 \\ m[1,2] + m[3,4] + P_0 P_2 P_4 & k=2 \\ m[1,3] + m[4,4] + P_0 P_3 P_4 & k=3 \end{cases}$$

$k=1$

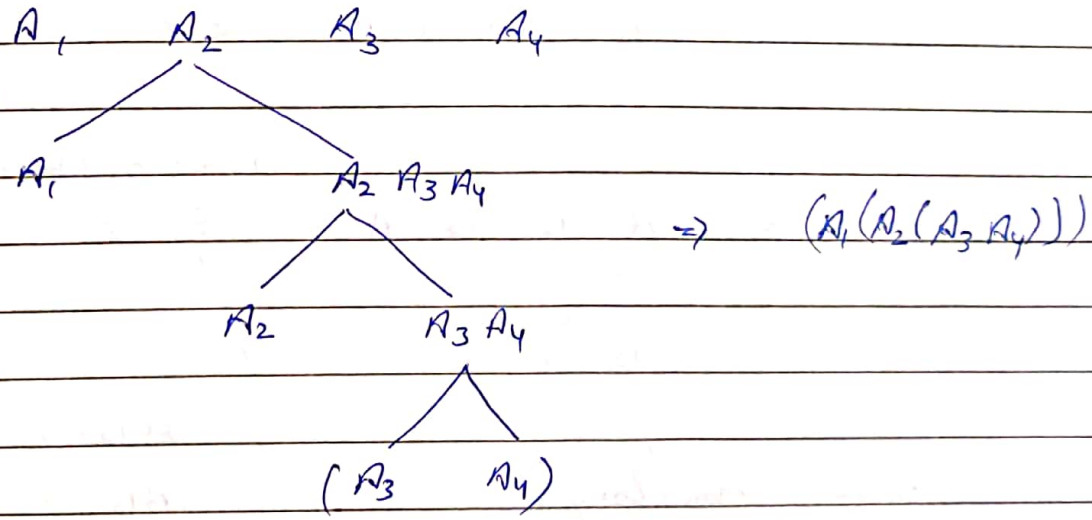
$k=2$

$k=3$

$$= \min \begin{cases} 0 + 10{,}000 + \overset{3000}{20 \times 30 \times 5} \\ 30{,}000 + 2500 + \overset{5000}{20 \times 50 \times 5} \\ 21{,}000 + 0 + \overset{1000}{20 \times 10 \times 5} \end{cases}$$

$$= \min \begin{cases} \cancel{10{,}000 + 3000 + 30} \\ \cancel{30{,}000 + 2500 + 5000} \end{cases}$$

$$= \min \begin{cases} 13000 \\ 37500 \\ 22000 \end{cases} = 13000$$



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

~~Longest Subsequence~~

Longest Common Subsequence (LCS)

We are given two sequences $X = \langle x_1, x_2 \ldots x_m \rangle$ and $Y = \langle y_1, y_2 \ldots y_n \rangle$ and wish to find maximum length common subsequence of X and Y.

Subsequence

X = IRONMAN

IRON
MAN
RONM
} Subsequence

ROIM ✗
MANI ✗

Common Sub sequence

X = < A B D A C E >
Y = < B A B C E >

BACE → Common Subsequence

X = < A B D A C E >
Y = < B A B C E >

ABCE → Common Subsequence
AB → "
CE → "
BCE → "

BACE
ABCE

Common/Longest Common Subsequence

$X = \langle A, B, C, B, D, A, B \rangle$

$Y = \langle B, D, C, A, B, A \rangle$

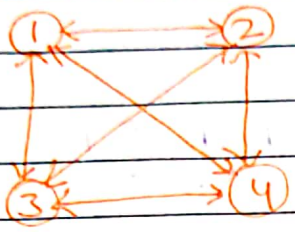
		0	1	2	3	4	5	6
		Y	B	D	C	A	B	A
0	X	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4 → Final LCS

Here, In last we get max/longest common subsequence = 4

~~A B C B~~ B C B A
4

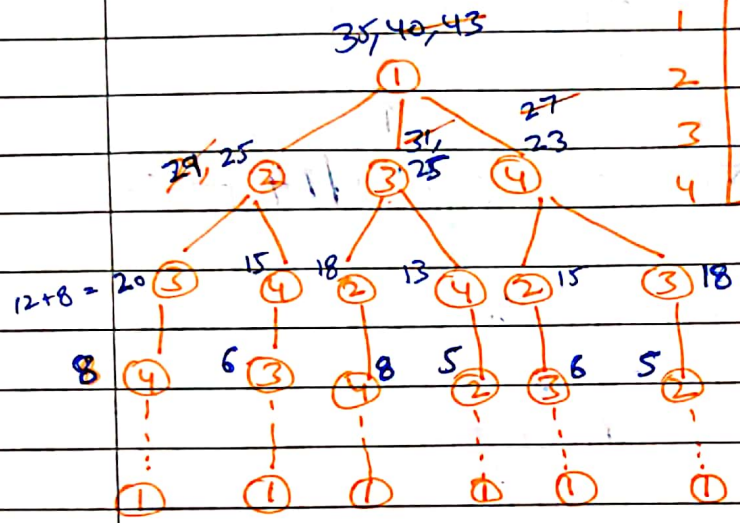
__/__/____

Travelling Salesman Problem



Start and End vertex should be same i.e. ~~Start~~ End to the same vertex from where started.
Ps: with minimum weight.

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



1-2-4-3-1 =

$$g(i,s) = \min_{J \in S} \{ c_{iJ} + g(J, S-\{J\}) \}$$

$g(i,s)$: is shortest path starting from 'i' and going through all vertex 'S' and terminate at vertex 'J'.

__/__/____

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

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

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

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

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

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

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

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

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

$$g(2, \{3,4\}) = \min \begin{cases} C_{23} + g(3, \{4\}) = 9 + 20 = 29 \\ C_{24} + g(4, \{3\}) = 10 + 15 = 25 \end{cases}$$

$= 25$

$$g(3, \{2,4\}) = \min \begin{cases} C_{32} + g(2, \{4\}) = 13 + 18 = 31 \\ C_{34} + g(4, \{2\}) = 12 + 13 = 25 \end{cases}$$

$= 25$

$$g(4, \{2,3\}) = \min \begin{cases} C_{42} + g(2, \{3\}) = 8 + 15 = 23 \\ C_{43} + g(3, \{2\}) = 9 + 18 = 27 \end{cases}$$

$= 23$

__/__/____

$$g(1,\{2,3,4\}) = \min \begin{cases} c_{12} + g(2,\{3,4\}) & = 10 + 25 = 35 \\ c_{13} + g(3,\{2,4\}) & = 15 + 25 = 40 \\ c_{14} + g(4,\{2,3\}) & = 20 + 23 = 43 \end{cases}$$

$= 35$

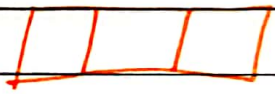
$1 \to 2 \to 4 \to 3 \to 1$

Backtracking

Backtracking uses brute force approach to solve problem. Brute force approach say that for any given problem generate all possible solution and pick up desired solution. Backtracking uses Depth-first-search to generate state space tree.

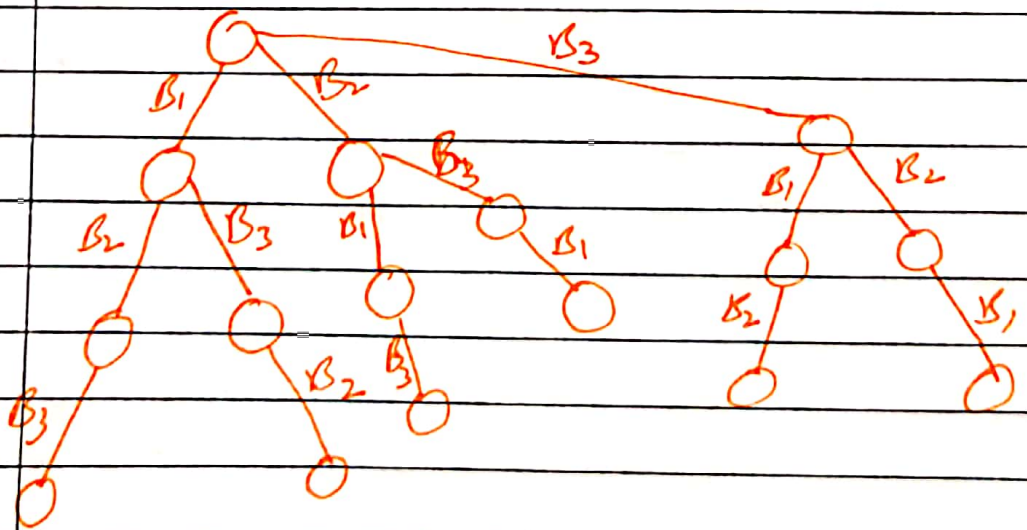
Eg: There are three students we have to arrange them in these three chairs.

B_1, B_2, B_3

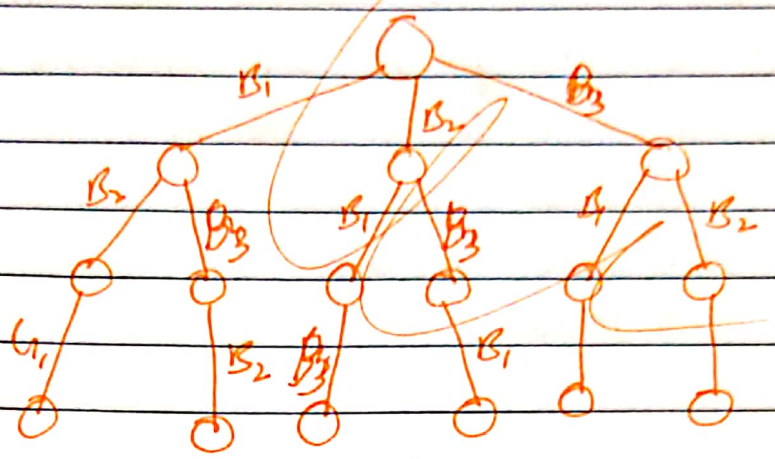


$n=3 \quad 3!$

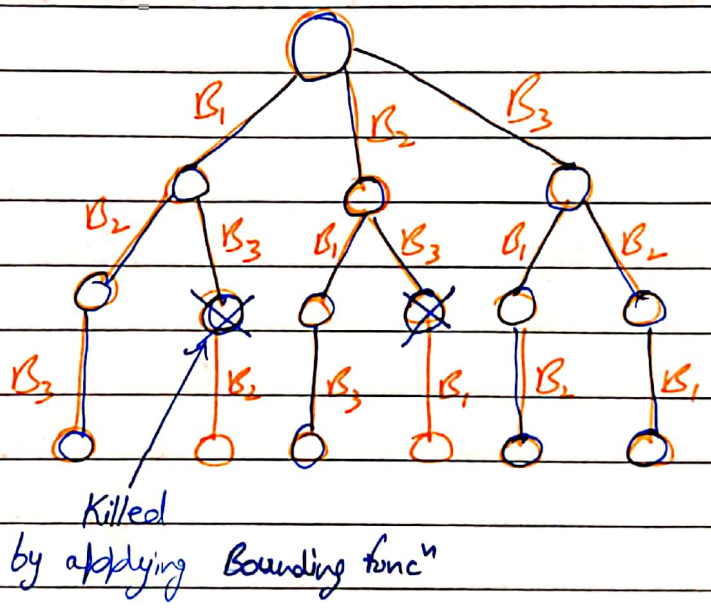
State space tree



Constraint : B_3 should not sit in between.



Constraint: B_3 should not sit in between.



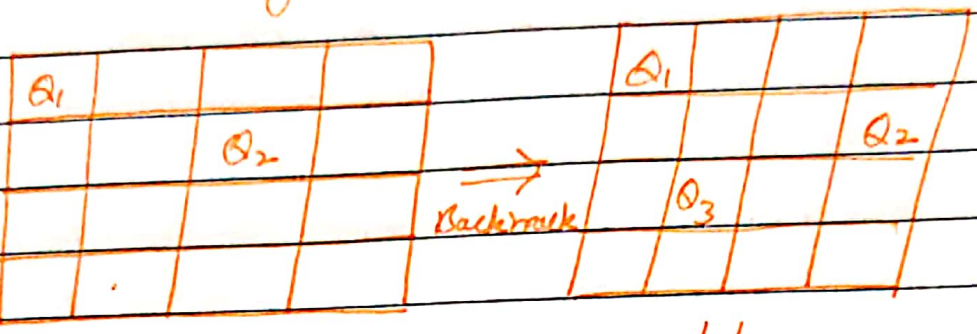
1) N Queen Problem
2) Sum of Subset Problem
3) Graph Coloring Problem

__/__/____

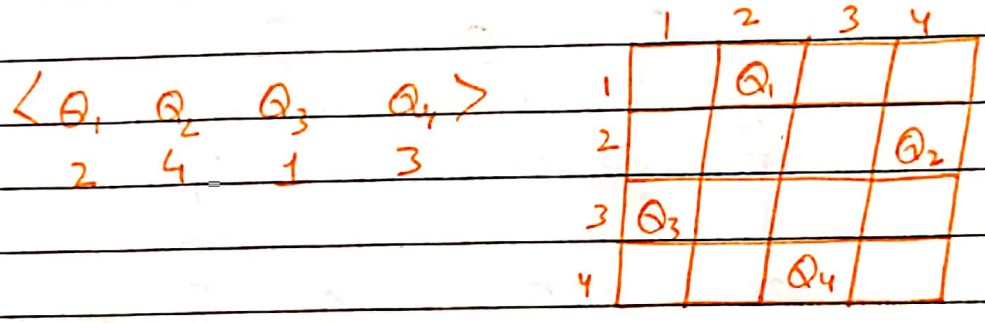
N Queen's Problem

N-Queens are to be placed in N×N chessboard so that no two Queens are attacking to each other. i.e no two Queen are in

1) Same row
2) Same column
3) Diagonal

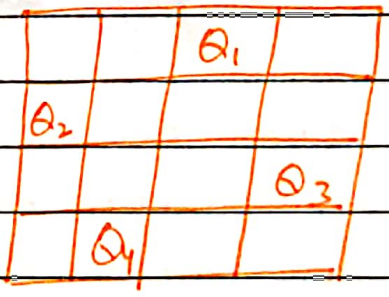


Backtrack

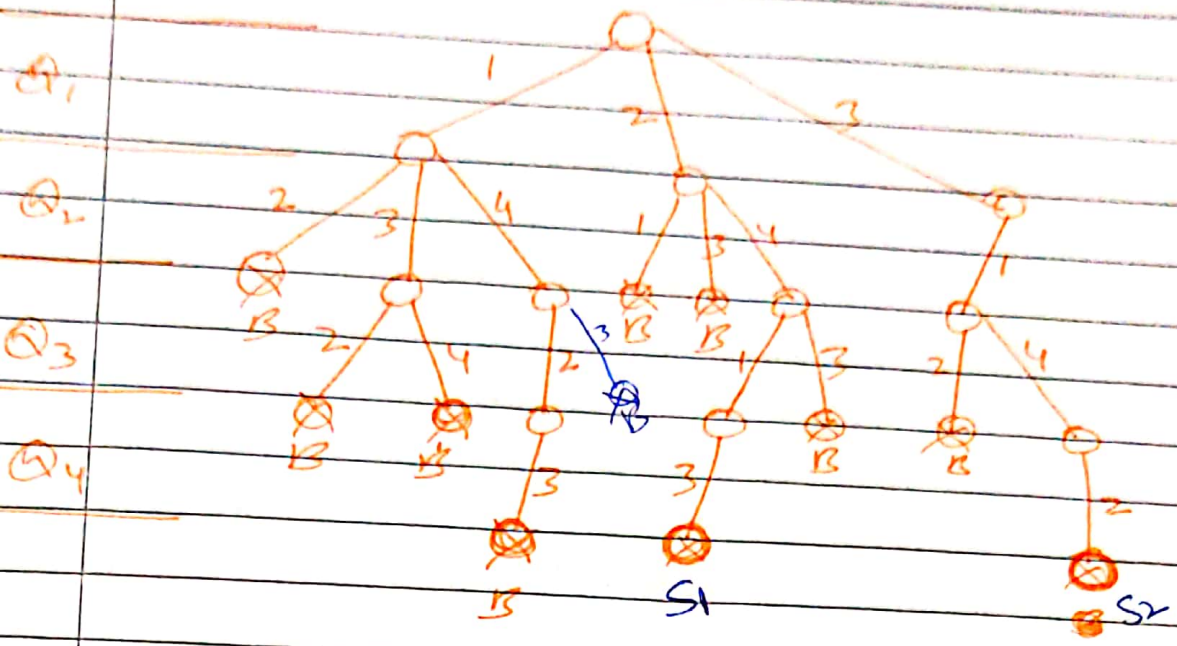


$\langle Q_1\ Q_2\ Q_3\ Q_4 \rangle$
2 4 1 3

mirror Sol.



$\langle Q_1\ Q_2\ Q_3\ Q_4 \rangle$
3 1 4 2



Sum of Subset

We are 'n' distinct positive integers and we desire to find all combination of those numbers whose sum are 'm'. This is called Sum of Subset.

Example: $n = 4$

$\{x_1, x_2, x_3, x_4\} = \{7, 11, 13, 24\}$

$m = 31$

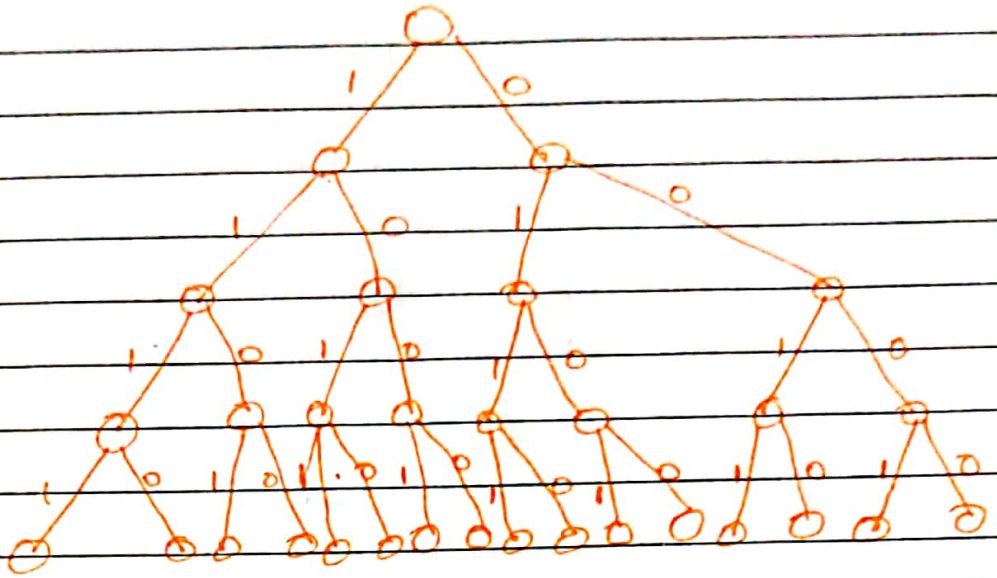
$\{7, 11, 13\}$ no duplicacy

$\{7, 24\}$ is allowed.

__/__/____

$\{x_1, x_2, x_3, x_4\}$ $n=4$

$= m$



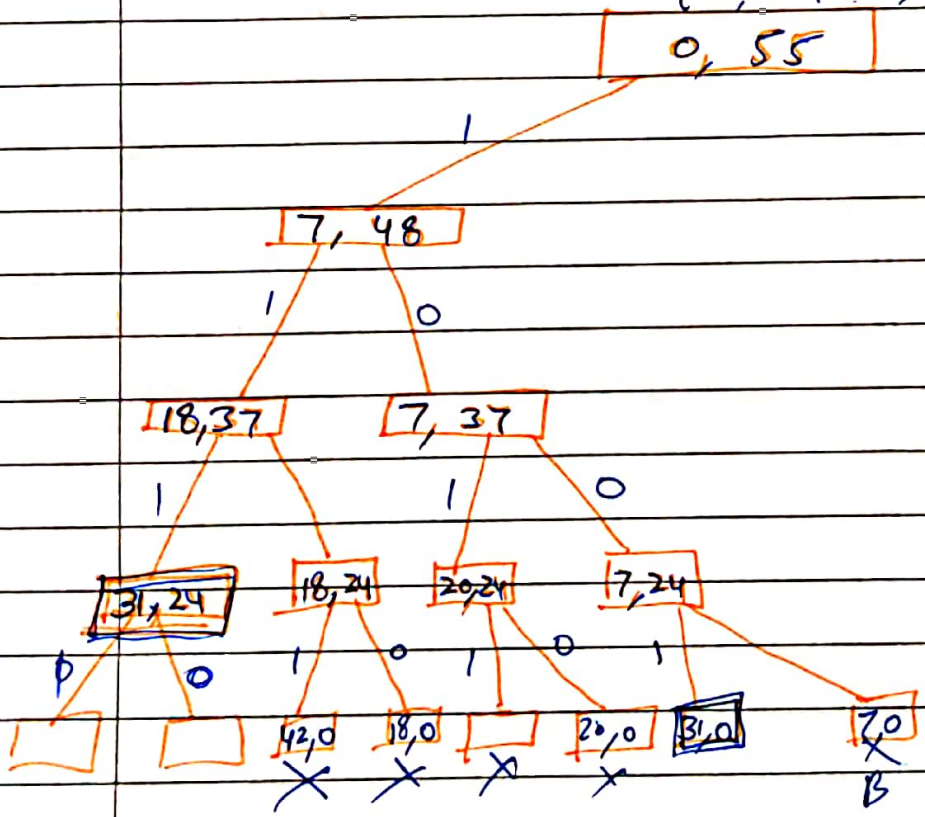
$n=4$

$m=31$

$\{x_1, x_2, x_3, x_4\} = \{7, 11, 13, 24\}$

$\{1, 1, 1, 0\}$

$\{1, 0, 0, 1\}$



__/__/____

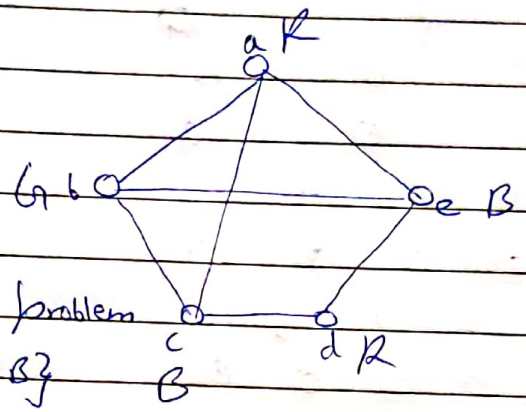
Graph Coloring Problem

Graph coloring is easy to color graph component such that vertex, edge & regions under some constraints.

No two adjacent vertices, edges or regions are colored with same color.

Vertex Coloring

$\{R, G, B\}$



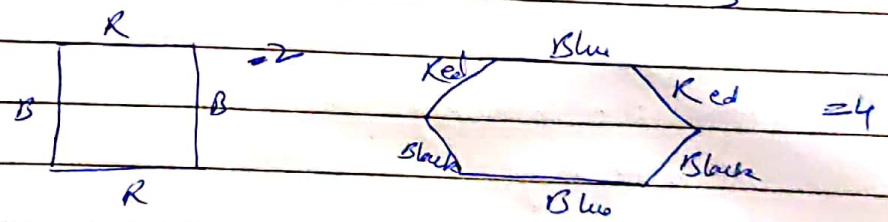
m-Coloring decision problem

$\{R, G\} \times \{R, G, B\}$

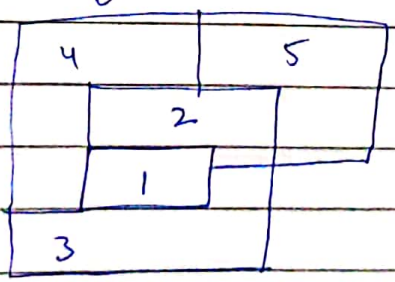
m-Coloring optimization problem

$\{R, G, B, Y\} \Rightarrow \{R, G, B\}^{K} \Rightarrow$ Chromatic problem $= 3$

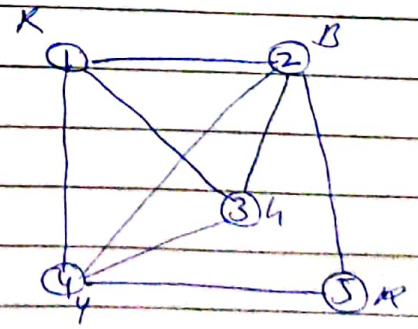
Edge Color

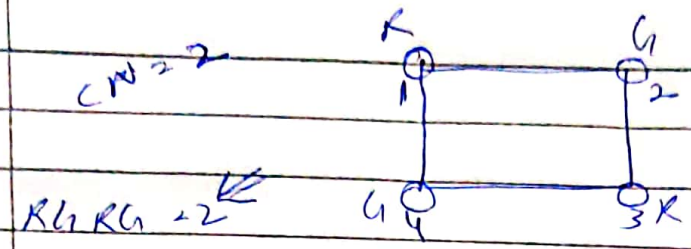


Region Coloring



$\Rightarrow 4$





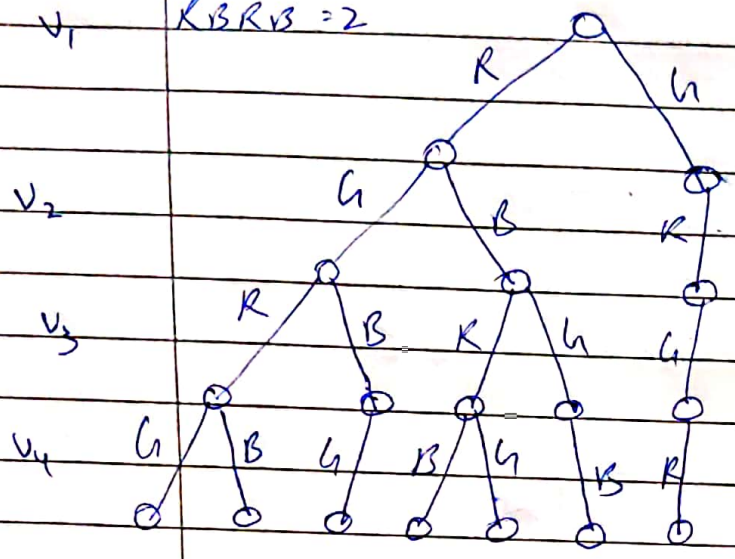
CN = 2

$m = \{R, G, B\}$

RG RG = 2

RG RB = 3

RB RB = 2



Sudoku

Time table Scheduling

Radio Frequency

Maps