* Job Sequencing Problem

Q]

jobs	J_1	J_2	J_3	J_4	J_5
Deadline	2	2	1	3	4
profits	20	60	40	100	80

→ 1) max deadline = 4

2) Arrange job in decreasing order of their profit

job	J_4	J_5	J_2	J_3	J_1
Deadline	3	4	2	1	2
profits	100	80	60	40	20 ×

J_3	J_2	J_4	J_5

0 1 2 3 4

(Gantt Chart)
max deadline is 4 hence we divide chart in 4 section.

profit = 40 + 60 + 100 + 80 = 280

Job sequence = $\{J_3, J_2, J_4, J_5\}$ Sequnce is ~~also~~ correct हुआ

Q] given the job & deadline & their profit are shown.

job	J_1	J_2	J_3	J_4	J_5	J_6
Deadline	5	3	3	2	4	2
profit	200	180	190	300	120	100

write a optimal shedule that gives the maximum profit, are all the job are completed in optimal shedule, what is maximum of all job?

→ 1) max deadline = 5

2) Arrange job in decreasing order of their profit

Job	J_4	J_1	J_3	J_2	J_5	J_6
deadline	2	5	3	3	4	2
profit	300	200	190	180	120	100

J_2	J_4	J_3	J_5	J_1

0 1 2 3 4 5

max profit = 180 + 300 + 190 + 120 + 200
= 990

optimal schedule = { J_2, J_4, J_3, J_5, J_1 }.

Not all Job are completed in optimal shedule

Q

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9
deadline	7	2	5	3	4	5	2	7	3
profit	15	20	30	18	18	10	23	16	25

→ 1) max deadline = 7
2) Arrange in decreasing order acc^n to profit

Job	J_3	J_9	J_7	J_2	J_4	J_5	J_8	J_1	J_6
deadline	5	3	2	2	3	4	7	7	5
profit	30	25	23	20	18	18	16	15	10

J_2	J_7	J_9	J_5	J_3	J_1	J_8

0 1 2 3 4 5 6 7

max profit = 20 + 23 + 25 + 18 + 30 + 15 + 16
= 147

Job sequence = { $J_2, J_7, J_9, J_5, J_3, J_1, J_8$ }

Huffman Coding

huffman coding lossless data compression algorith
The idea is to assigned to variable length code to i/p characters length of the assign code are based on frequency of corresponding charact
[the variable length code assign to i/p character are prefix code means that codes are assigned in a such way that a code assign to one character is not the prefix of code assigne to any other character] The code length of character dependes on how the frequently it occurs in the given text. the character which occurs must frequently get the smallest code & the character which occurs less frequently get the largest code

- There two major step in huffman coding
1) building a huffman tree from the i/p characts
2) assigning code to the character by traversing the half man tree.

Steps involve to construct huffman tree

1) create a leaf node of each char of text leaf node of char contain the occuring freq of that character

2) Arrange all the node in increasing order of their freq value

3) conside the first 2 node having minimum freq. create new internal node the freq of these new node is the sum of freq. of these two nodes make the first node as left child & other node as right child of newly created node.

4] If the repeating step 2 & 3 until all the nodes form a single tree, tree finally obtain is the desire haffman tree.

- average code length per character = $\dfrac{\sum \text{freq} \times \text{code length}}{\sum \text{freq}}$

- total No. of beats in huffman encoded msg = Total No. of character in msg $\times$ avg code length per character

(1) fixed size coding (2) variable size code

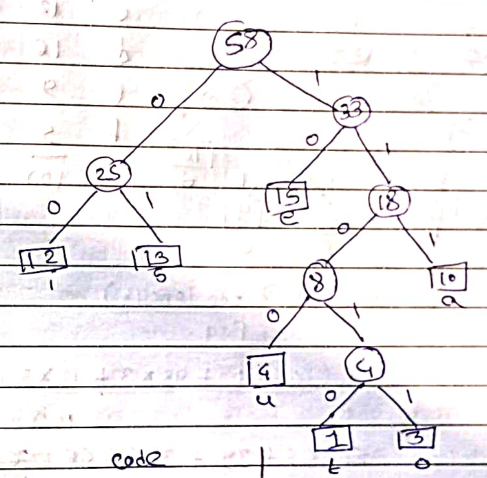
Q. A file contain the foll^n character with a freq as shown

char	freq	
a	10	33
e	15	25
i	12	18
o	3	8
u	4	
s	13	
t	1	4

If haffman coding is use for data compression then determine
1) Haffman code for each character
2) avg code length
3) length of encoded msg in beat

Consider 2 min freq

→



1]

		code
a	10	111
e	15	10
i	12	00
o	3	11011
u	4	1100
s	13	01
t	1	11010
	58	

2] avg code length = $\dfrac{\sum(\text{freq} \times \text{code length})}{\sum \text{freq}}$

$= \dfrac{10\times3 + 15\times2 + 12\times2 + 3\times5 + 4\times4 + 13\times2 + 1\times5}{58}$

$= 2.52 \approx 2.5$

3) Formula :- Total length = Avg code length $\times$ total no. of freq

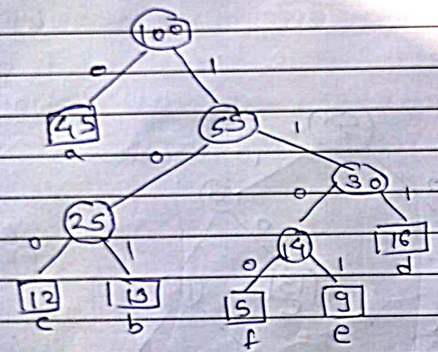
$= 2.52 \times 58$

$= 146.16$ bit is required for sending msg

Q a b c d e f — find total No. of bit required for sending msg.
45 13 12 16 9 5

suppose we have 10^5 character in data bite file, find out how many bits are required to sending msg.

		code
a	45	0
b	13	101
c	12	100
d	16	111
e	9	1101
f	5	1100
	100	



1) avg code length $= \frac{\Sigma(\text{freq} \times \text{code length})}{\Sigma \text{freq}}$

$= \Sigma \frac{(45\times1 + 13\times3 + 12\times3 + 16\times3 + 9\times4 + 5\times4)}{100}$

$= \Sigma(45 + 39 + 36 + 48 + 36 + 20)$

$= \frac{224}{100} = 2.24 \times 10^5$

~~Total length = Avg code length + total No. of freq.~~

~~= 224 + 100~~

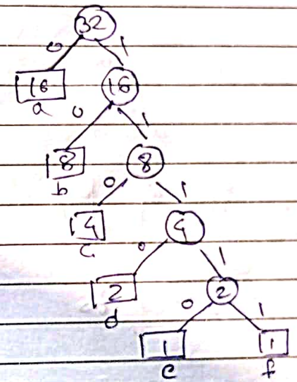
~~= 2.24~~

$\therefore$ $2.24 \times 10^5 \times 100$ bits are req^n to send msg

$= 224 \times 10^5$

Q a b c d e f
1/2 1/4 1/8 1/16 1/32 1/32

		multiply by 32	code
a	1/2	16	0
b	1/4	8	10
c	1/8	4	110
d	1/16	2	1110
e	1/32	1	11110
f	1/32	1	11111
		32	



1) avg code length $= \frac{\Sigma(\text{freq} \times \text{code length})}{\Sigma \text{freq.}}$

$= \Sigma \frac{(16\times1 + 8\times2 + 4\times3 + 2\times4 + 1\times5 + 1\times5)}{32}$

$= \frac{(16+16+12+8+5+5)}{32} = \frac{62}{32} = 1.93$

2) Total length = Avg code length × total No. of freq

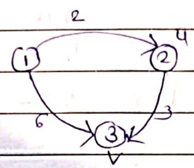
$= 1.93 \times 32$

$= 61.76 \simeq 62$

Krishna
DATE: __.__.__

* Dijkstra Algorithm
- use for to find out shortest path

* Relaxation of an edge method.

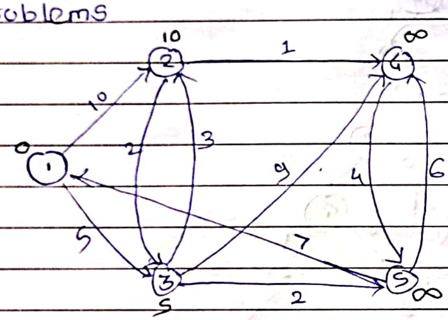


$d(v) > d(u) + \omega(u,v)$

$d(v) = d(u) + \omega(u,v)$

* Problems

Q

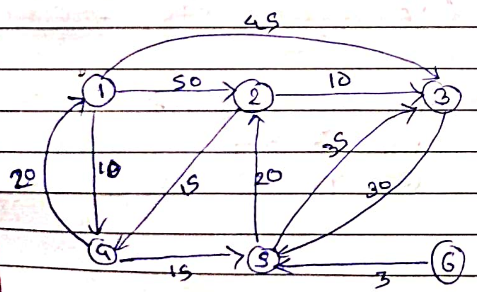


vertex	vertices	2	3	4	5
3	{1, 3}	10	5	∞	∞
5	{1, 3, 5}	8	5	14	7
2	{1, 3, 5, 2}	8	5	13	7
4	{1, 3, 5, 2, 4}	8	5	9	7

∴ cost fin.

2 → 8
3 → 5
4 → 9
5 → 7

Q



Krishna
DATE: __.__.__

Algorithm

Dijkstra (G, ω, S)
Initialize single source (G, S)
S ← Φ — O(V)
Q ← V[4] — O(V)
while Q ≠ Φ
do u ← Extract min(Q) — v log v
S ← S ∪ {u} — 1
for each vertex v ∈ Adj[u] — v^2 i.e E log V
do
Relax (u, v, ω)

* Initialize single source (G, S)
for each vertex $V \in V[G]$
do $d[V] \leftarrow \infty$
pred[V] = Nil
$d[S] \leftarrow 0$

Signature

Krishna
DATE: __.__.__

Relax (u, v, w)
if (d[v] > d[u] + w(u,v))
then
d[v] = d[u] + w(u,v)
pred[v] ← u

complexity of algo → O(V) + O[V] + V log V + E log V + 1

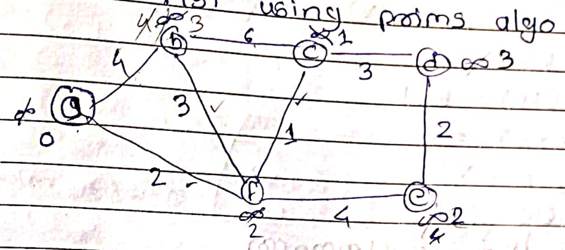
→ O[(V+E) log V]
or
→ O[E log V]

limitation:-
If there is -ve weight is present then these algorithm sometime ~~not~~ work. & sometime not work.

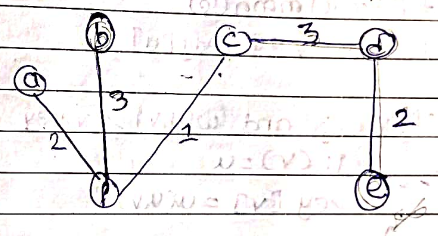
* Prims Algo
steps :-
1] Starting with one any vertex & change the d[v] from ∞ to 0
2] then choose the child edges & compare there weight, which has minimum weight
3] before 2 step apply u, v formula i.e
(u, v) = w(u, v) < d[v]
d[v] = w(u, v)
4] after step 3 again follow step 2 for next vertex which are choose in 2 step... so on
5] also check the child edge & previous remaining edge

Krishna
DATE: __.__.__

Q find the MST using prims algo



→



	a	b	c	d	e	f
key	0	3	1	3	2	2
parent	N	f	f	c	d	a

Signature ______

Algorithm:- graph, weight graph, starting point

$V \leq$ or E

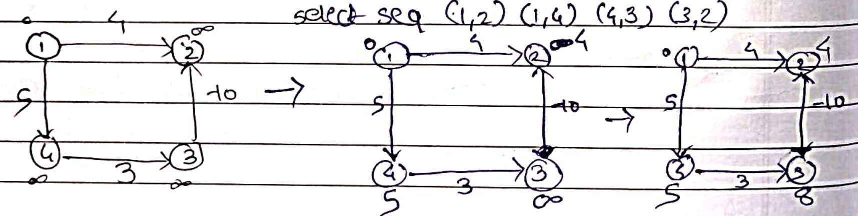
prims (G, w, r)
for each $u \in G(V)$
$u.key = \infty$
$\pi(u) = Nil$
key $(r) = 0$
$Q = G(V)$ — $O(V)$
while $Q \neq \phi$ .... ($Q = \{a, b, c, d \ldots$ vertex$\}$) — $O(V)$
u = Extractmean (Q) — $O(\log v)$
for each $v \in G.Adj[u]$ — $O(V)$
{
if $v \in Q$ and $w(u,v) < v.key$
{
$\pi(v) = u$ — $O(1)$
key $[v] = w(u,v)$ — $O(\log v)$
}
}

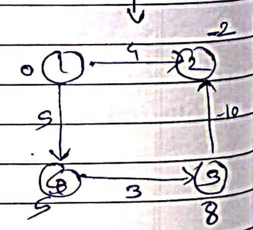
complexity $\begin{cases} (V+E)\log V \\ \text{or} \\ V \log V \end{cases}$

* Bellman ford Algorithum

select seq (1,2) (1,4) (4,3) (3,2)

Q



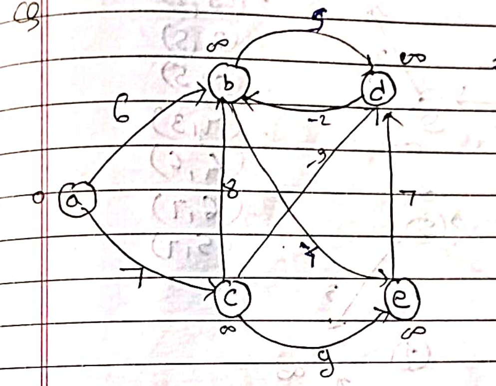


* if V vertices the $|V-1|$ time relaxing edge method ~~reduce~~ / doing this step
$|4-1| = 3$ times

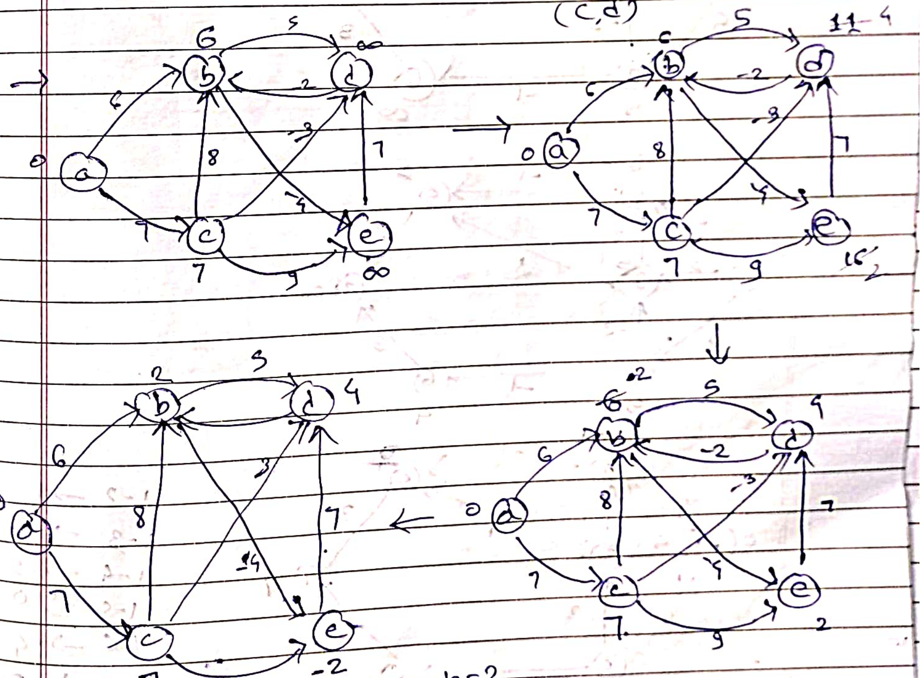
* update condition:-
$d(v) > d(u) + w(u,v)$
$d(v) = d(u) + w(u,v)$

complexity is greater than disaster Algo Here we use Bellman & if -ve edge is present then also Bellman use.

Q

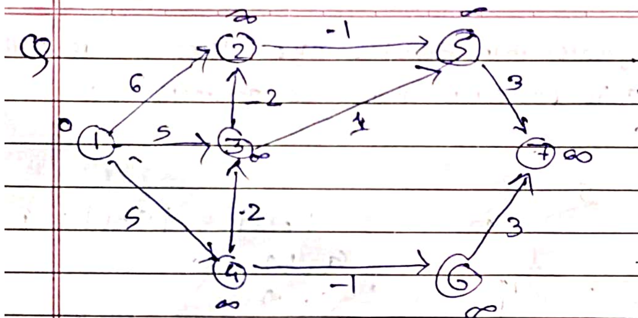


$\Rightarrow |V-1| = |5-1| = 4$
(a,b)
(a,c)
(c,b)
(b,d)
(c,e)
(e,d)
(d,b)
(b,e)
(c,d)

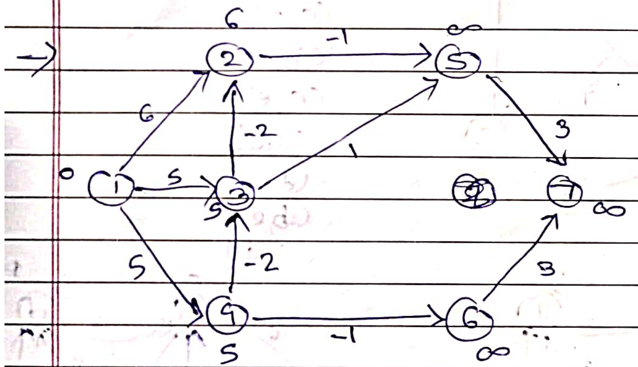


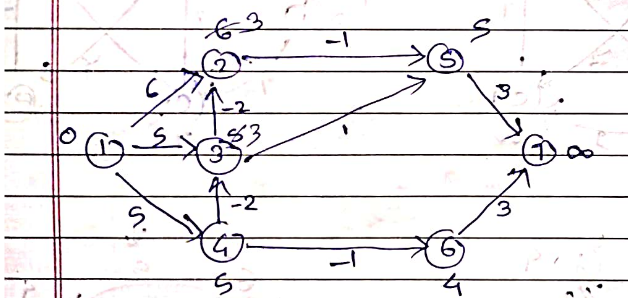
Ans ⇒ a-b = 2
a-c = 7
a-d = 4
a-e = -2

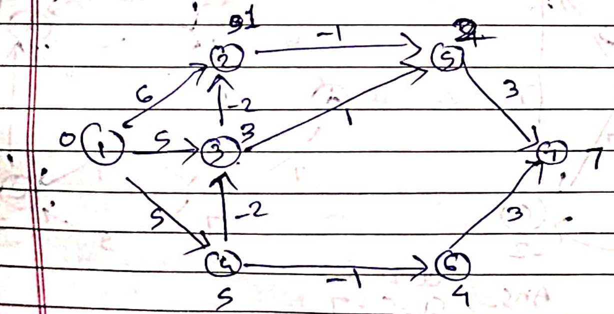
Q



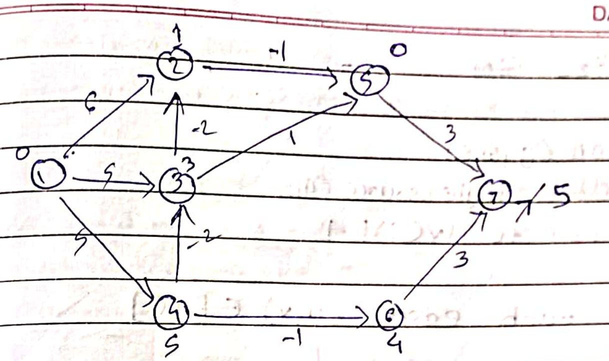
(1,2)
(1,4)
(1,3)
(3,2)
(2,5)
(3,5)
(4,3)
(4,6)
(6,7)
(5,7)

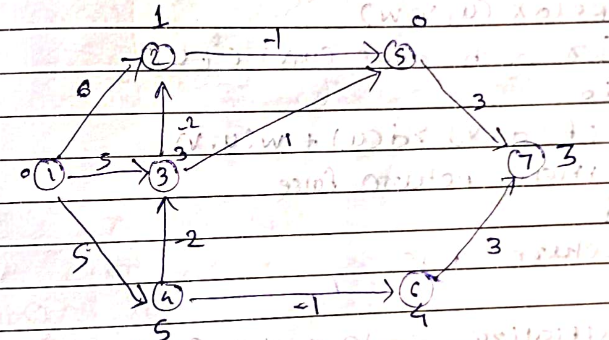


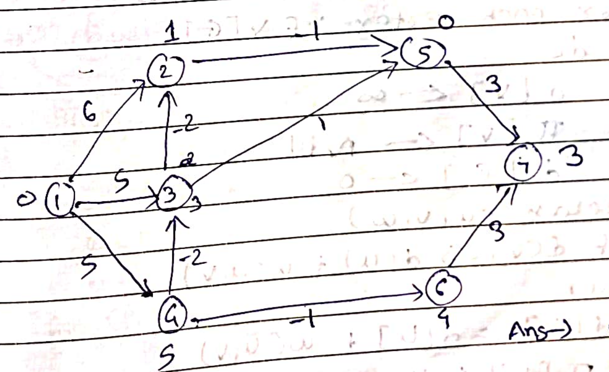




1-2 → 1
1-3 → 3
1-4 → 5
1-5 → 0
1-6 → 4
1-7 → 3







Ans →
1-2 → 1
1-3 → 3
1-4 → 5
1-5 → 0
1-6 → 4
1-7 → 3

Krishna
DATE: __.__.__

Initialize Sing

Bellman (G,W,S)
Initialize single source (G,S)
for i=1 to |V(G)|-1
do
for each edge (u,v) ∈ E(G]
do
Relax (u,v,w)
For each edge (u,v) ∈ E[G]
do
if d(v) > d(u) + w(u,v)
then Return false
}
Return true.

Initialize single source (G,S)
for each vertex v ∈ V[G]
do
d [v] ← ∞
π [v] ← Nill
d [G] ← 0
Relax (u,v,w)
if d(v) -> d(u) + w(u,v)
then
d[v] = d[u] + w(u,v)
π [v] ← u

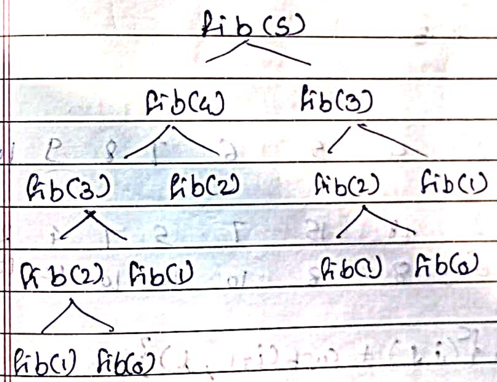
Time complexity O(V*E)

Signature

Krishna
DATE: __.__.__

Dynamic Programming
1) Principle of optimality -> dividing a big problem & Breaking it into subproblem g taking decision at every stage.
2) overlapping subproblem
3) optimal substructure.
4) memorization

$$fib(n) = \begin{cases} fib(0) & \text{if } n=0 \\ fib(1) & \text{if } n=1 \\ fib(n-1)+fib(n-2) & \text{if } n \geq 2 \end{cases}$$



n=5 function called for 15 times

$\frac{2^n}{2}$ minimizing the extremely large complexity

buffer memory.

0	1	1	2	3
fib(0)	fib(1)	fib(2)	fib(3)	fib(4)

```
int fib(n) {
  fib[0]=0
  fib[1]=1
  for (int i=2; i<n; i++)
  {
    fib[i] = fib(i-1) + fib(i-2);
  }
  return fib[n]
```

fib[5] = 5

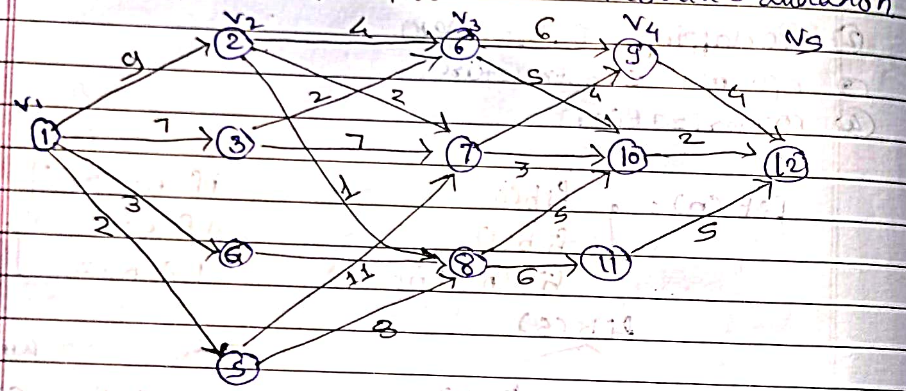
fib[n] function is being called for n+1 time.

Signature

Krishna
DATE: __.__.__

- Applications
* for optimization prob :

* multistage Graph Problem :- Resource allocation



	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

$cost(i,j) = \min\{c(j,l) + cost(i+1, l)\}$
$<j, l> \in E$
$l \in V_{i+1}$

- start from sink, sink → source

stage 5 cost (5,12) = 0

stage 4
cost (4,9) = 4
cost (4,10) = 2
cost (4,11) = 5

stage 3 cost (3,6) = min(c(6,9) + cost(4,9), ⇒ 6+4=10
c(6,10) + cost(4,10)} 5+2=7
min (10,7) = 7

Krishna
DATE: __.__.__

prepare strategy or intelligence gathered following process
- Stackholder identification
- Stackholder analysing.
- stackholder matrix
- " engagement
- " attribute.

---x---x---

cost (3,7) = min(c(7,9) + cost (4,9),
c(7,10) + cost (4,10))
= min (4+4, 3+2)
= min (8,5)
= 5 from 7 to 10

cost (3,8) = min{c(8,10) + cost (4,10),
c(8,11) + cost(4,11) }
= min (5+2, 6+5)
= min (7,11)
= 7 from 8 to 10

stage 2 cost (2,2) = min = {c(2,6) + cost (3,6),
c(2,7) + cost (3,7),
c(2,8) + cost (3,8) }
= min{4+7, 2+5, 1+7}
= min {11,7,8}
= 7 from 2 to 7

Signature ____

cost (2,3) = { c(3,7) + cost (3,7),
c (3,6) + cost (3,6) }
= { 7+5,
2+7 }
= 9 from 3 to 6

cost (2,4) = min {c(4,8) + cost (3,8)}
= 11 + 7
= 18 from 4 to 8

cost (2,5) = min (c(5,7) + cost(3,7), c(5,8)+cost(3,8))
= (11+5, 8+7)
= 16, 15
= 15 from 5 to 8

stage 1 =)

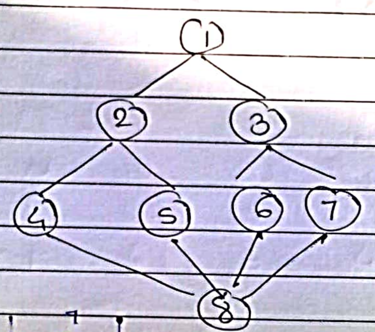
cost (1,1) = min { c(1,2) + cost(2,2)
c (1,3) + cost (2,3)
c (1,4) + cost (2,4)
c (1,5) + cost (2,5)
= min { 9+7, 7+9, 3+18, 2+15 }
= 16 r = 2,3

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

1-2-7-10-12 1-3-6-10-12

Krishna
DATE: __.__.__

* Depth first search → use stack.



1	1	~~0~~1	~~0~~1	1	1	1	1
1	2	3	4	5	6	7	8

initially it is 0 & when visited it become 1

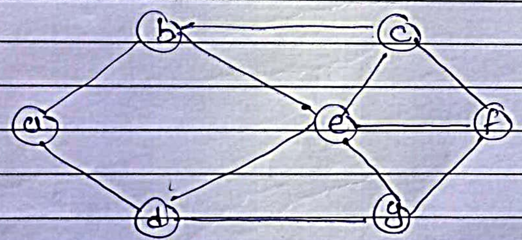
~~adjacent~~
adjacent

u=1 → (2,3)
u=2 → (1,4,5)
~~u=3~~ → 6
u=4 → (2,8)
u=8 → (4,5,6,7)
u=5 → (2,8)
u=6 → (3,8)
u=3 → (1,6,7)
u=7 → (3,8)

Stack: 7, 3, 6, ~~5~~, 8, 4, 2, 1

O/P → 1, 2, 4, 8, 5, 6, 3, 7

Ex - 1)



Stack: d, g, f, e, c, b, a

a → (b,d)
b → (c,e,a)
c → (b,e,f)
e → (b,c,f,g,a)
f → c, e, g)
g → (f, e, d)
d → (a, e, g)

a, b, c, e, f, g, d

Krishna
DATE: __.__.__

Algorithm :-
DFS(V,E)
for each v∈V
mark[v] = 0
for each v∈V
if (mark [v]=0) then
dfs(v)

dfs(v)
mark[v]=1
print v
for each (v,u) ∈ Adj[v]
if (mark [u]=0)
dfs(u)

Time complexity :-
1) Adjacency matrix
[ ] n×n → in this we require 2 for loop for traversing
∴ $O(V^2)$

2) Adjacency list

[a] → [b] → [d]
[b] → [a] → [c] → [e]

In this vertex is at least connect to 1 vertex
→ 2V
And we make one array to make visited 1
→ E

O(V+E)

Krishna
DATE: __.__.__

* 0/1 knapsack prob
↳ solve by greedy as well as dynamic

m=50

W	10	20	30	000	– 0
P	60	100	120	001	– 120
P	6	5	4	010	– 100
				100	– 60
				101	– 180
				110	– 160
				011	– 220
				111	– X

$TC = 2^n$
↓
minimizes

→ using dynamic programming:

* $KS(n,w) = \begin{cases} 0 & n=0, w=0 \\ KS(n-1, w) \;\;(\text{case II}) & wt(n) > W \\ \max \begin{cases} KS(n-1, w) \\ KS(n-1, (W - wt(n)) + P(n)) \end{cases} \;\;(\text{case III}) \end{cases}$

Q: W=7, n=4
profit = {1, 2, 5, 6} W = {2, 3, 4, 5}

	W	P	w=0	w=1	w=2	w=3	w=4	w=5	w=6	w=7
n=0			0	0	0	0	0	0	0	0
n=1	1	1	0	1	1	1	1	1	1	1
n=2	3	4	0	1	1	4	5	5	5	5
n=3	4	5	0	1	1	4	5	6	6	9
n=4	5	7	0	1	1	4	5	7	8	9

Krishna
DATE: __.__.__

$KS(1,1) = 1$

$KS(1,2) = \max \begin{cases} KS(0,2) \\ KS(0,(2-1)) + 1 \end{cases}$
$(0,1)+1$
$0+1$

$= 1$

KS same for
(1,3) (1,4) (1,5) (1,6)
(1,7)

$KS(2,1) = KS(n-1, w)$ — case I wt(n)>w
$= KS(1,1)$

$KS(2,2) = KS(n-1, w)$
$= KS(1,2)$

$KS(2,3) = \max \begin{cases} KS(n-1, w) \\ KS(n-1, w - wt(n)) + (n) \end{cases}$

$= \max \begin{cases} KS(1,3) \\ KS(1,(3-3)+4) \end{cases}$
$0+4$

$= 4$

$KS(2,4) = \max \begin{cases} KS(n-1, w) \\ KS(n-1, w - wt(n)) + n \end{cases}$

$= \begin{cases} KS(1,4) \\ KS(1,(4-3)) + 4 \end{cases}$

$= KS(1,1) + 4$
$1 + 4$

$= 5$

$KS(2,5) = \begin{cases} KS(1,5) \\ KS(1,(5-3)) + 4 \end{cases}$
$KS(1,2)+4$
$1+4$

$= 5$

Signature

Krishna

DATE: __.__.__

Case II $KS(3,1) = \{ KS(2,1)$

$KS(3,2) = \{ KS(n-1, w)$

$= \{ KS(2,2)$

$= 1$

$KS(3,3) = \{ KS(2,3)$

$= 4$

$KS(3,4) = max \begin{cases} KS(n-1, w) \\ KS(n-1, w-wt(n)) + P \end{cases}$

$= \begin{cases} KS(2,4) \\ KS(2,(4-4)) + 5 \end{cases}$

$KS(2,0)$

$0 + 5$

$= 5$

$KS(3,5) = max \begin{cases} KS(2,5) = 5 \\ KS(2,(5-4)) + 5 \end{cases}$

$= max \begin{cases} 5 \\ 1 + 5 \end{cases}$

$= 6$

$KS(3,6) = max \begin{cases} KS(2,6) \\ KS(2,(6-4)) + 5 \end{cases}$

$= max \begin{cases} 5 \\ 1+5 \end{cases}$

$= 6$

$KS(2,7) = max \begin{cases} KS(2,7) \\ KS(2,(7-4)) + 5 \end{cases}$

$4+5=9$

$= \begin{cases} 5 \\ 9 \end{cases}$

$KS(4,1) = \{ KS(3,1) = 1$

$KS(4,2) = KS(3,2) = 1$

$KS(4,3) = KS(3,3) = 4$

$KS(4,4) = KS(3,4) = 5$

$KS(4,5) = max \begin{cases} KS(3,5) \\ KS(3,(5-5)) + 7 \end{cases}$

$= \begin{cases} 6 \\ 0+7 \end{cases}$

$= 7$

$KS(4,6) = max \begin{cases} KS(3,6) \\ KS(3,(6-5)) + 7 \end{cases}$

$= \begin{cases} 6 \\ 1+7 \end{cases}$

$= 8$

$KS(4,7) = \begin{cases} KS(3,7) \\ KS(3,(7-5)) + 7 \end{cases}$

$= \begin{cases} 9 \\ 8 \end{cases}$

$= 9$

Sol^n set $\Rightarrow$

x_1	x_2	x_3	x_4
0	1	1	0

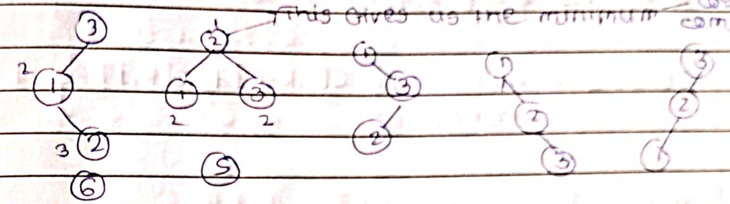
Krishna

DATE: __.__.__

- Optimal Binary Search Tree (OBST)

1, 2, 3 → key element

This gives us the minimum cost comparison

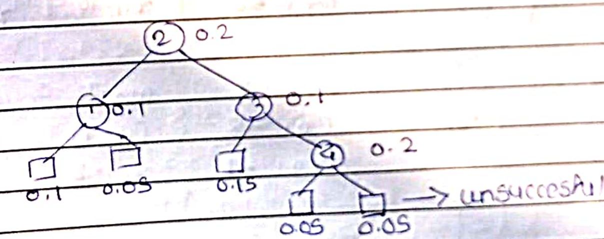


Formula $T(n) = \frac{2nC_n}{n+1}$ $n = 3$

$\frac{2 \times 3C_3}{3+1} = \frac{6C_3}{4} = \frac{6!}{(6-3)! \, 4}$

$= 5$

key	1	2	3	4	
p	0.1	0.2	0.1	0.2	successful
q 0.1	0.05	0.15	0.05	0.05	unsuccessful



$= 1 \times 0.2 + 2 \times 0.1 + 2 \times 0.1 + 3 \times 0.1 + 3 \times 0.05 + 3 \times 0.15 + 3 \times 0.2 + 4 \times 0.05 + 4 \times 0.05$

$= 2.5$

optimal Binary search tree are those who give minimal searching result

for this we use dynamic programming.

Signature ____

3 3 1 1   2 3 1 1

Krishna

DATE:__.__.__

keys = {1, 2, 3, 4}

p_i = {3, 3, 2, 1}

q_i = {2, 3, 1, 1, 1}

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

$$w[i,j] = w[i,j-1] + p_j + q_j$$

w - weight

c - cost

r - root node

	0	1	2	3	4
	$w_{00}=2$	$w_{11}-3$	$w_{22}-1$	$w_{33}=1$	$w_{44}=1$
$j-i=0$	$c_{00}=0$	$c_{11}=0$	$c_{22}=0$	$c_{33}=0$	$c_{33}=0$
	$r_{00}=0$	$r_{11}=0$	$r_{22}=0$	$r_{33}=0$	$r_{33}=0$
	$w_{01}-8$	$w_{12}-7$	$w_{23}-3$	$w_{34}-3$	
$j-i=1$	$c_{01}-8$	$c_{12}-7$	$c_{23}-3$	$c_{34}-9$	
	$r_{01}-1$	$r_{12}-2$	$r_{23}-3$	$r_{34}-4$	
	$w_{02}-12$	$w_{13}-9$	$w_{24}-5$		
$j-i=2$	$c_{02}-19$	$c_{13}-12$	$c_{24}-8$		
	$r_{02}-1$	$r_{13}-2$	$r_{24}-3$		
$j-i=3$	$w_{03}-14$	$w_{14}-11$			
	$c_{03}-25$	$c_{14}-19$			
	$r_{03}-2$	$r_{14}-2$			
$j-i=4$	$w_{04}-16$				
	$c_{04}-32$				
	$r_{04}-2$				

Signature___

Krishna

DATE:__.__.__

$w[0,1] = w[0,0] + p_1 + q_1$

$= 2 + 3 + 3$

$= 8$

$w[1,2] = w[1,1] + p_2 + q_2$

$= 3 + 3 + 1 = 7$

$w[2,3] = w[2,2] + p_3 + q_3$

$= 1 + 1 + 1 = 3$

$w[3,4] = w[3,3] + p_4 + q_4 = 1+1+1 = 3$

$w[0,2] = w[0,1] + p_2 + q_2 = 8+3+1 = 12$

$w[1,3] = w[1,2] + p_3 + p_3 = 7+1+1 = 9$

$w[2,4] = w[2,3] + p_4 + p_4 = 3+1+1 = 5$

$w[0,3] = w[0,2] + p_3 + q_3 = 12+1+1 = 14$

$w[1,4] = w[1,3] + p_4 + q_4 = 9+1+1 = 11$

$w[0,4] = w[0,3] + p_4 + q_4 = 14+1+1 = 16$

$$c[0,1] = \min_{i<k\le j} \{ c[0,0] + c[1,1] + w[0,1]$$

$0 < 1$, 1

$= 0 + 0 + 8$

$= 8$

$$c[0,2] = \min_{0<k\le 2} \begin{cases} c[0,0] + c[1,2] + w[0,2] \\ c[0,1] + c[2,2] + w[0,2] \end{cases}$$

(1, 2)

$\begin{cases} 0 + 7 + 12 \\ 8 + 0 + 12 \end{cases}$

$\min\begin{cases}19\\20\end{cases} \to 19 - k = 1$

$$c[1,3] = \min_{1 \le k \le 3} \left\{ \begin{matrix} c[1,1] + c[2,3] \\ 0 + 3 \\ c[1,2] + c[3,3] \\ 7 + 0 \end{matrix} \right\} + w[1,3]$$

2, 3

$\min\begin{cases}3+9\\7+9\end{cases} = 12$ $k = 2$

Signature___

Krishna
DATE: __.__.__

$$C[2,4] = \min_{\substack{2<k\le 4 \\ 3,4}} \left\{ \begin{array}{l} C[2,2]+[3,4] \\ 0+3 \\ C[2,3]+C[4,4] \\ 3+0 \end{array} \right\} + \omega[2,4]$$

$$\min \begin{array}{l} 3+5 \\ 3+5 \end{array} = 8 \quad k=3$$

$$C[0,3] = \min_{\substack{0<k\le 3 \\ 1,2,3}} \left\{ \begin{array}{l} C[0,0]+C[1,3] \\ C[0,1]+1[2,3] \\ C[0,2]+[3,3] \end{array} \right\} + \omega(0,3)$$

$$\left\{ \begin{array}{l} 0+12 \\ 8+13=11 \\ 19+0 \end{array} \right\} + 14 = 25 \quad k=2$$

$$C[1,4] = \min_{\substack{1<k\le 4 \\ 2,3,4}} \left\{ \begin{array}{l} C[1,1]+C[2,4] \\ C[1,2]+C[3,4] \\ C[1,3]+C[4,4] \end{array} \right. \quad \omega[1,4]$$

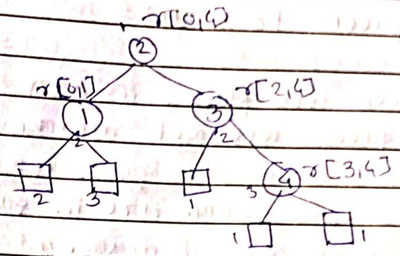
$$= \min \left\{ \begin{array}{l} 0+8 \\ 7+3 \\ 12+0 \end{array} \right\} + 11 = 19$$

$$C[0,4] = \min_{\substack{0<k\le 4 \\ 1,2,3,4}} \left\{ \begin{array}{l} C[0,0]+C[1,4] \\ C[0,1]+C[2,4] \\ C[0,2]+C[3,4] \\ C[0,3]+C[4,4] \end{array} \right. + \omega[0,4]$$

$$= \left\{ \begin{array}{l} 0+19 \\ 8+8 \\ 19+3 \\ 25+0 \end{array} \right. + 16 = 16+16 = 32 \quad k=2$$

Signature____

Krishna
DATE: __.__.__



$r(i,j) = k$

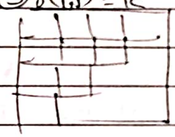
$r(i,k-1) \quad r(k,j)$

steps ① ~~form~~ $C(i,j)=c(i,k-1)+c(k,j)$ $+\,\omega(i,j)$

② $\omega(i,j)=\omega(i,j-1)+p(j)+q(j)$

③ $r(i,j)=k$

steps ②



step ③ construct tree

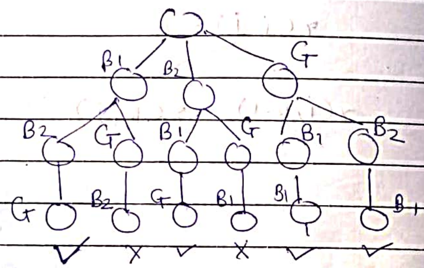
Signature____

Krishna
DATE: __.__.__

- N-queen Problem

N queen are placed in n×n chessboard then queens are not attacking to ~~and~~ each other.
i.e no two queen are in same row, same column & diagonal!

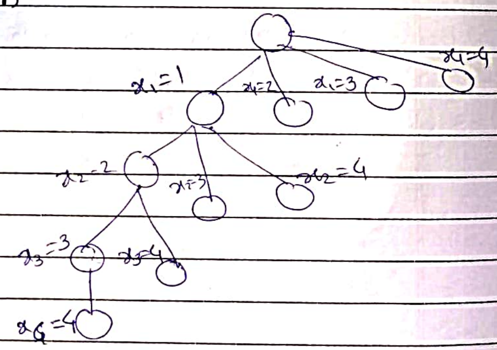
| B1 | B2 | G | →



G cannot sit in middle
→ 4 solution

- 4 queen problem

	Q1		
			Q2
Q3			
		Q4	



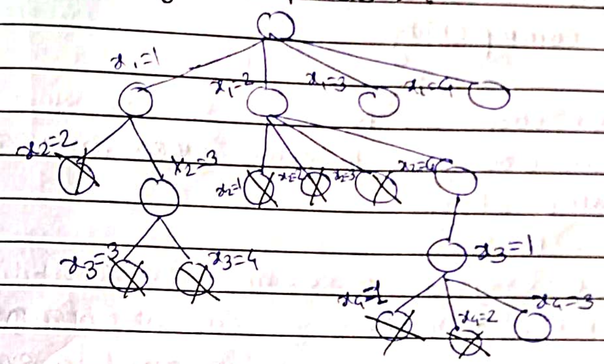
| 2 | 4 | 1 | 3 |

In this there are many possibilities that dose not satisfy our problem - we have to minimize it

$= 1\times4 + 4\times3 + 4\times3\times2 + 4\times3\times2\times1$
$= 4 + 12 + 24 + 24$
$= 64$

Krishna
DATE: __.__.__

minimizing the problem :-



Algorithm

```
Algorithm Nqueen (k,n)   -> queen
{
  for i=1 to n do
  {
    if place (k,i) then
    {
      x[k] = i;
      if (k = n) then
        write (x [1:n]);
      else
        nqueen (k+1, n)
    }
  }

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
}
```

Signature ______

Krishna

DATE: __.__.__

* Time complexity

worst case → N!
- as we are try to place queen in every column of row
- ex. N=4
Row 1 : try all 4 column (4 possibility
Row 2 : 3 possibility (as we place Q1 in row1 in same column)
Row 3 : 2 possibility
Row 4 : 1

∴ 4 x 3 x 2 x 1 = 24
N! = 4! = 4 x 3 x 2 x 1 = 24

Algo :-

```
bool issafe (int row, int column) {
  for (i=0 ; r < row ; i++) {
    if (board [i] = col || abs (board [i] - col) =
         same column
                         = abs (i - row))
                              same diagonal
       return false
    }
    return true;
}
```

• sum of subset problem

$S = \{10, 20, 30, 40\}$ (with x_1, x_2, x_3, x_4 above the elements)

M = 50

{20, 30}　　|0|1|1|0|

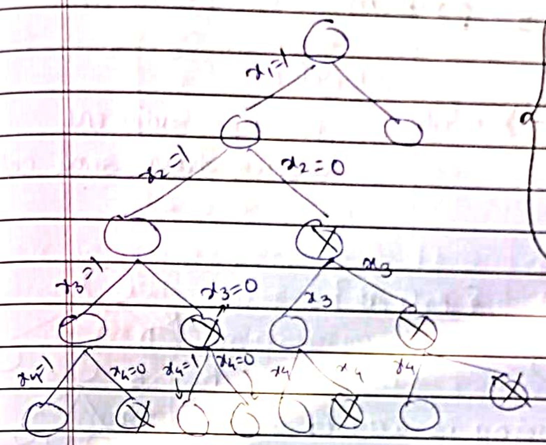
{10, 40}

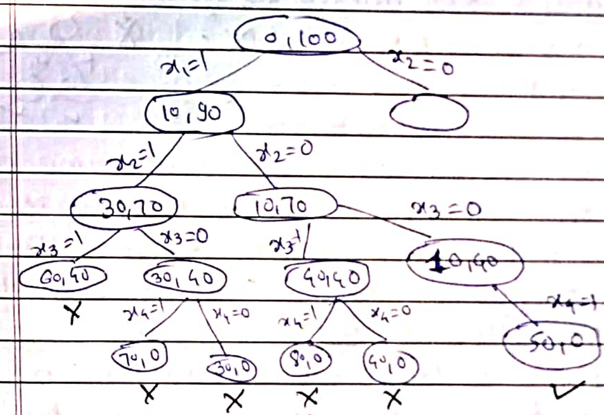
Signature____

Krishna

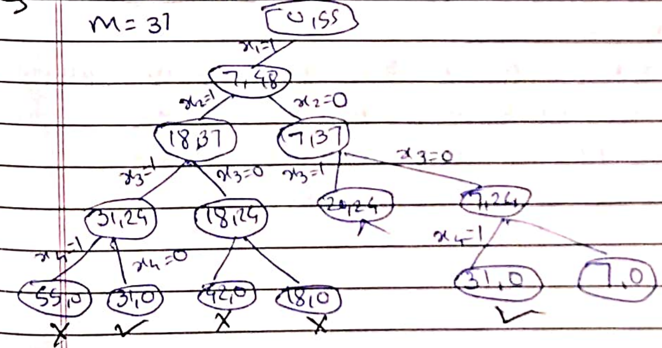
DATE: __.__.__

$$\left\{ \begin{array}{l} \text{i) } \sum_{i=1}^{k} w_i x_i + w_{k+1} \le m \\ \sum_{i=1}^{k} w_i x_i + \sum_{i=k+1}^{n} \ge m \end{array} \right.$$



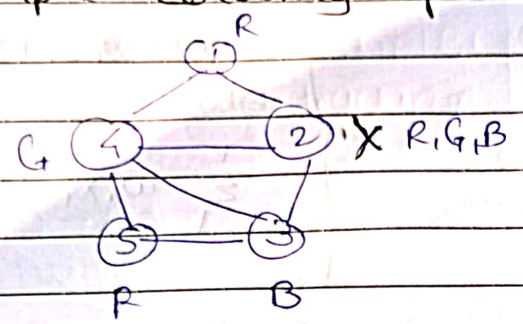


Q.2 S={7,11,13,24}
M= 31



Signature____

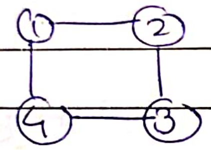
* graph coloring problem



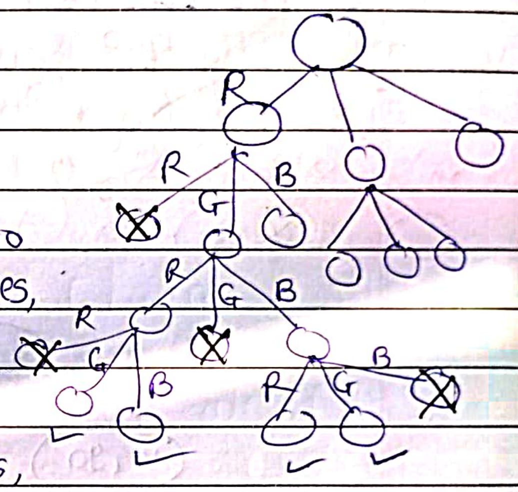
{R, G, B}

No two adjacent vertex have same color

ex -



{R, G, B}



- The graph coloring is way to color graph such that vertices, edges & region under some constraint
- like no 2 adjacent vertices, edges or region are color with same color.
- if graph is given & some colors are given we want to know whether graph can be color by using those color or not this problem called as M coloring decision prob.
- if graph is given & we want to know minimum how many color require for coloring graph called as M-coloring optimization prob.
- if graph is given & color is given then we want to know how many possible way graph can be color is called graph coloring problem.

backtracking (DFS)

② To find all optimal soln (to get every soln) ⇒ use bound (BFS)

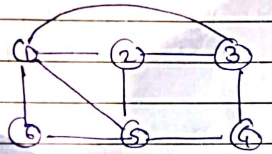
Krishna

DATE: __.__.__

* Hamilatonion cycle

graph may directed or not but must be connected

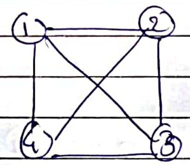
- There are multiple Hamilatonion cycle is possible by using backtracking



cycle : 1 2 3 4 5 6 1
1 6 5 4 3 2 1

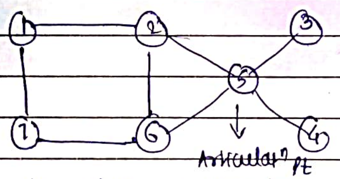
- If there are n no. of vertices present in graph then formation of hamilatonion cycle is $\frac{(n-1)!}{2}$

ex.



1 2 3 4 1
1 4 3 2 1
2 3 4 1 2 (duplicate of 1st)
2 4 3 1 2

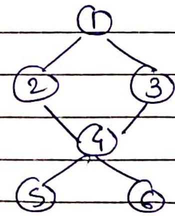
ex.



If articulation point is present in graph then hamiltonion cycle is not formed because it take double the give vertices.

~~the give vertices~~ i.e here 5 is coming twice

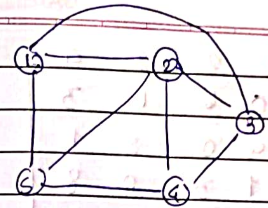
ex.



If pendent node is present in graph then also hamilatonion cycle will not be formed

DATE: __.__.__

ex



Adj matrix

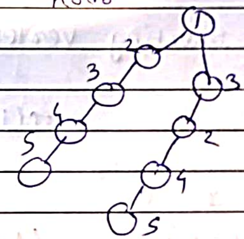
	1	2	3	4	5
1	0	1	1	0	1
2	1	0	1	1	1
3	1	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

must have direct edge with previous

1 2 3 4 5 → direct edge from last to first node

1st case | 0 | 0 | 0 | 0 | 0 |

2nd case | 1 | 3 | 2 | 4 | 5 |



3. bounding fn.

1) we should not take a duplicate cycle
2) whenever we take vertex then it should be edge with previous verties
3) If we are not last vertex then it must have edge with first vertex

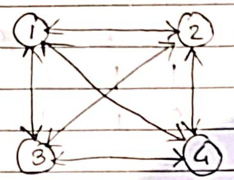
* Travelling Salesman Problem.

1) minimal cost reqrd to travel the graph
2) we can solve travelling salaman problem by dynamic prog & branch & bound problem

Signature

by dynamic prog.

ex:



	1	2	3	4
1	0	6	1	3
2	4	0	2	1
3	1	3	0	8
4	3	1	7	0

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

i = starting vertex

T(1, {2,3,4}) [T] min of all 3

6 + 9 = 15   1 + 6 = 7   3 + 4 = 7

(1,2) + T(2,{3,4})   (1,3) + T(3,{2,4})   (1,4) + T(4,{2,3})

min

[2] [4] = 13 [1] [8] = 9   2 + [4] = 6   8 + 5 = 13   1 + 3 = 4

(2,3) + T(3,{4})   (2,4) + T(4,3)   (3,2) + T(2,4)   (3,4) + T(4,2)   (4,2) + T(2,3)   (4,3) + T(3,2)

[3] [1] [1]

(3,4) + T(4,∅)   (4,3) + T(3,∅)   (2,4) + T(4,∅)   (4,2) + T(2,∅)   (2,3) + T(3,∅)   (3,2) + T(2,∅)

[8]   [1] [3]   [1] [4]   [2] [1]   [3] [4]

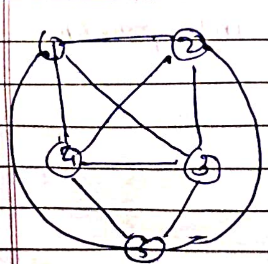
consider 1

1,3,2,4

1,4,2,3

by branch & bound method

ex.



	1	2	3	4	5	small of value of row
1	∞	20	30	10	11	10
2	15	∞	16	4	2	2
3	3	5	∞	2	4	3
4	19	6	18	∞	3	4
5	16	4	7	16	∞	21

21 - all rows   3 - all colm

∞	1	-9	11	10
6	∞	5	17	19
18	16	∞	19	17
2	15	-3	∞	18
5	17	14	5	∞

∞	8	0	-2	-1
3	∞	4	-8	-10
-9	-7	∞	-10	-8
7	-6	6	∞	-9
4	-8	-5	4	∞

smallest value in col   2   1   -9   5   10 = 9

Consider starting vertex as 1 & first row & col be ∞

C(1,2)

~~(2,1) → ∞~~ because we have

consider C(1,2)

∞	∞	∞	∞	∞	
∞	∞	4	-8	-10	-10
-9	∞	∞	-10	-8	-10
7	∞	6	∞	-9	-9
4	∞	-5	4	∞	-5
					-34