Q. $f(n) = n^2 + 10$ Find the upper bound.

→ condn for Big O notation

$F(n) \leq c \cdot g(n)$ where $n \geq n_0$, c, n_0 are const

$F(n) = n^2 + 10$ assume $g(n) = (2)n^2$ → c.

$n^2 + 10 \leq 2n^2$ $c = 2$.

	$F(n)$	$g(n)$
	$n^2 + 10$	$2n^2$
$n = 1$	11	2
$n = 3$	19	18
$n = 4$	26	32
$n = 5$	35	50

$n^2 + 10 \leq 2n^2$ is true

for $c = 2$ & $n_0 = 4$

$n^2 + 10 = O(n^2)$.

Q. $F(n) = 5n + 50$

→ condn for Big O notation

$F(n) \leq c \cdot g(n)$ $n \geq n_0$, c, n_0 is constant

$F(n) = 5n + 50$ assume $c = 10$

$5n + 50 \leq 10n$

	$F(n)$	$g(n)$
	$5n + 50$	$10n$
$n = 1$	55	10
$n = 10$	100	100
$n = 11$	105	110

} it create a tight bound.

$5n + 50 \leq 10n$ is true for $c = 10$ & $n_0 = 11$

~~$5n + 10$~~ $5n + 50 = O(n)$

For $c = 6$. $n_0 = 50$. $5n + 50 \leq 6n$

	$5n+50$	$6n$
$n = 49$	295	294
$n = 50$	300	300
$n = 51$		

Q. $F(n) = 5n + 50$ $g(n) = \log_{10} n$ is $g(n)$ upper bound of $F(n)$.

→ condn for Big O notation

$F(n) \leq c.g(n)$.

	$F(n)$	$g(n)$
	$5n+50$	$100.\log_{10} n$
$n = 10$	100	100
$n = 11$	105	104.
$n = 100$	550	200
$n = 1000$	5050	300

the condn is since $F(n) \not\leq c\,g(n)$

Q. $F(n) = 2^n + 3n^3$ Find out upper bound.

$g(n) = 2 \cdot 2^n = 2^{n+1}$

↳ c.

	$F(n)$	$g(n)$
	$2^n + 3n^3$	2^{n+1}
$n = 5$	407	64
$n = 1$	5	4.
$n = 2$	28	8.
$n = 13$	14783	16.384.

$2^n + 3n^3 \leq 2^{n+1}$ For $c = 2$

$n_0 = 13$.

dominating term consider as $g(n)$ is lower bound

Q. Find lower bound of $F(n) = 10n^2 + 5$.

→ consider the Big Ω notation

$F(n) \geq c.g(n)$ $\quad n \geq n_0 . \; c \geq 0 . \; n \geq 1$

$F(n) = 10n^2 + 5$ assume $g(n) = n^2$ $\quad 10n^2$

$10n^2 + 5 \geq n^2$

	$10n^2 + 5$	$10n^2$
$n = 1$	15	10
$n = 2$	45	40

$10n^2 + 5 = \Omega(n^2)$

$c = 10 . \; n = 1$

Q. show that $F(n) = n^3 + 3n^2 = \Theta(n^3)$

→

$c_1 . g(n) \leq F(n) \leq c_2 . g(n)$ $\quad g(n) = n^3$

	$n^3 \leq n^3 + 3n^2 \leq 2 . n^3$
$n=1$	$1 \leq 4 \leq 2$
$n=2$	$8 \leq 20 \leq 16$
$n=3$	$27 \leq 54 \leq 54$ $\quad n = 3$

$c_1 g(n) \leq F(n) \leq c_2 g(n)$

$n^3 \leq n^3 + 3n^2 \leq 2n^3$ true for

$c_1 = 1 . \; c_2 = 2 \; n >= 3 .$

$F(n) = \Theta(g(n))$

$n^3 + 3n^2 = \Theta(n^3)$

- $n^3 + 3n^2 = O(n^2)$ for $c = 2$ & $n_0 = 3$

$F(n) \geq c_1 . g(n)$

$F(n) = n^3 + 3n^2$

Assume $g(n) = n^3$, $c = 1$

$n^3 + 3n^2 \geq n^3$?

$n^3 + 3n^2 = \Omega(n^3)$

for $c = 1$ & $n_0 = 1$

Condition Θ notation

$c_1 . g(n) \leq F(n) \leq c_2 . g(n)$

For $c_1 = 1$, $n_0 = 3$ $c_2 = 2$

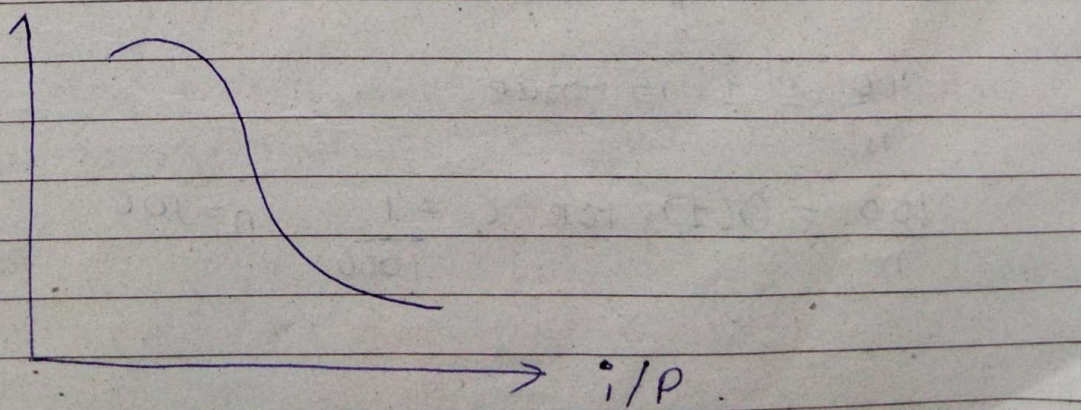
* Decreement Function.

a function which has ~~m~~ numerator value is less than the denominator.

$\frac{10}{n}$ $\frac{n}{n^2}$ $\frac{n}{2^n}$, $\frac{n^4}{5^n}$, $\frac{n^5}{5^n}$

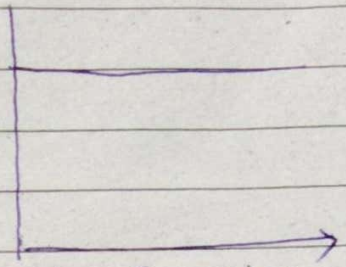
$= \frac{10}{n}, \frac{1}{n}, \frac{n}{2^n}, \frac{n^4}{5^n}, \frac{n^5}{6^n}$

$= \frac{10}{n}, \frac{1}{n}, \frac{n}{2^n}, \frac{n^5}{5^n}, \frac{n^4}{5^n}$



* Constant function

ex - 100, 2^{10}, 5, 1 thousand.



constant function is asymptotically greater or bigger than decrement function.

e.g. $F(n) = \frac{100}{n}$ $g(n) = 1000$

is $F(n) = O(g(n))$

→ $F(n) \leq C \cdot g(n)$ where $n \geq n_0$ & C & n_0 are constant

$F(n) = \frac{100}{n}$ $C = \frac{1}{1000}$ $C g(n) = \frac{1}{1000} \times 1000$

$\frac{100}{n} \leq 1 = ?$

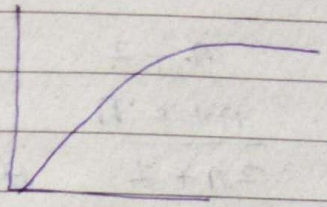
	$F(n)$ $\frac{100}{n}$	$g(n)$ 1
$n = 10$	10	1
$n = 100$	1	1
$n = 1000$	0.1	1
$n = 10000$	0.01	1

$\frac{100}{n} \leq 1$ is true

$\frac{100}{n} = O(1)$ for $C = \frac{1}{1000}$, $n = 100$

• logarithmic Function

eg. $\log n$ $(\log n)^5$ -- $\log\log n$, $\log n \cdot \log\log n$



ex) $F(n) = 100$, $g(n) = \log_{10} n$

→ $F(n) \leq c \cdot g(n)$

$F(n) = 100$, $g(n) = \log_{10} n$

	F(n)	g(n)
	100	$\log_{10} n$
$n = 10$	100	1
$n = 10^{100}$	100	100
$n = 10^{1000}$	100	1000

logarithmic Funn value is greater than constant Funn

* polynomial function.

ex. n^2, $\sqrt{n}$, n^{10}, n^k

$F(n) = \log_{10} n$, $g(n) = \sqrt{n}$

is $F(n) = 0$ $g(n) = ?$

$F(n) \leq c \cdot g(n)$

	F(n)	g(n)
	$\log_{10} n$	$\sqrt{n}$
$n = 10$	1	3.16
$n = 100$	2	10

* Exponential Function

ex. 2^n, $n!$, n^n, 5^n

exponential Funn is bigger than the polynomial Funn

$F(n) = \sqrt{n}$, $g(n) = 2^n$

	F(n)	g(n)
$n = 2$	$\sqrt{n}$	2^n
$n = 4$	1.41	4
	2	16

Decrement Funn < constant Funn < logarithmic Funn < polynomial Funn < exponential Funn

* problem.

* Time Complexity

1) For (i=1; i<=n ; i++)
   printF ("computer")

$n+1$
n
$2n+1$ $\quad \Theta(n)$

2) Algo sum (a, n)
{
Sum = 0; — 1 unit times
for (i=1; i<=n ; i++) — n+1
Sum = Sum + a[i]; — n
return Sum; — 1
}

$2n+3$ units
$\Theta(n)$

3) For (i=1; i<=n ; i+2)
{
pf ("computer");
}

i = 1, 1+2, 1+2x2, 1+3x2 ---- 1+Kx2
1, 3, 5, 7

Assume $1+Kx2 = n$
$K = \frac{n-1}{2}$

$\Theta(n)$.

4) for (i=1; i<=n; i+5)
{
pf ('c')

i = 1, 1+5, 1+2x5, 1+3x5 -- 1+Kx5

Assume $1+Kx5 = n$
$K = \frac{n-1}{5}$

$\Theta(n)$.

for all ($i*2$, $i/2$) increment in for loop.
the complexity is $\log n$ $i*5$, $i*7$ 5,7 does not matter.

5) for (i=n ; i>=1 ; i--)
pf ("comp")

$i=n$ $n-1$ $n-2$ $n-3$ ---- $n-K$

$n-K=1$

$K=n-1$

$\Theta(n)$

6) For (i=n ; i>=1 ; i-2)
pf ("comp")

$i=n$ $n-2^1$ $n-2^1*2$ $n-2*3$ --- $n-2K$.

Assume $n-2K=1$

$K=\frac{n-1}{2}$

$\Theta(n)$.

7) For (i=1 ; i<=n ; i*2)
pf ("c");

$i=1$ $1*2$ 2^2 2^3 ---- 2^K

Assume $2^K=n$

$\log_2 2^K = \log_2 n$

$K = \log_2 n$

$\Theta(\log n)$.

8) for (i=n ; i>=1 ; i/2)
pf

~~i=1~~ ~~n/2~~ $2/2$ _ $K/2$

$\frac{K}{2}$

$i=n$ $\frac{n}{2}$ $\frac{n}{2^2}$ $\frac{n}{2^3}$ --- $\frac{n}{2^K}$.

$\frac{n}{2^K}=1 \rightarrow 2^K=n \rightarrow K=\log n$ - $\Theta(\log n)$

9) For ($i=3$; $i<=n$; $i=i^2$)

$a=a+5$

	1	2	3		K
$i=3$	3^2	3^2	3^2	...	3^{2^K}

Assume $3^{2^K} = n$

$\log_3$ on both side

$\log_3 3^{2^K} = \log_3 n$

$2^K \log_3 3 = \log_3 n$

$2^K = \log_3 n$

apply $\log_2$ on both side

$\log_2 2^K = \log_2 \log_3 n$

$K \log_2 2 = \log_2 \log_3 n$

$K = \log_2 \log_3 n$

$\downarrow$

$\Theta(\log \log n)$

10) For ($i=n$; $i>=5$; $i=\sqrt{i}$)

		$\frac{1}{2}$	$\frac{1}{2^2}$	$\frac{1}{2^3}$	$\frac{1}{2^K}$
$i=n$	~~$\sqrt{n}$~~	n	n	n	... (n)

Assume $(n)^{\frac{1}{2^K}} = 5$

Apply $\log_5$ on b.s

$\log_5 n^{\frac{1}{2^K}} = \log_5 5$

$\frac{1}{2^K} \log_5 n = 1$

$2^K = \log_5 n$

$\log_2 2^K = \log_2 \log_5 n$

$K \log_2 2 = \log_2 \log_5 n$

$K = \log_2 \log_5 n$

$\downarrow$

$\Theta(\log \log n)$

11) For (i=1; i<=2^n; i++)

i=1 i+1 1+2 1+3 ... 1+K.

$1+K = 2^n$

$K = 2^n - 1$

↓

$\Theta(2^n)$

$1+K = \sqrt{n}$

$K = \sqrt{n}$

$\Theta(\sqrt{n})$

12) For (i=1; i<=$\sqrt{n}$; i++)

↓

$\Theta(\sqrt{n})$

13) For (i=n^2; i>=1; i = i/2)

pf ("comp");

$i = n^2$ $\frac{n^2}{2}$ $\frac{n^2}{2^2}$ ... $\frac{n^2}{2^K}$

$\frac{n^2}{2^K} = 1$

$n^2 = 2^K$

$\log_2 n^2 = K \log_2 2$

$K = 2\log_2 n$

↓

$\Theta(\log n)$

14) For (i=5^n; i>=5; i=$\sqrt[5]{i}$)

$i = 5^n$ $(5^n)^{\frac{1}{5}}$ $(5^n)^{\frac{1}{5^2}}$ ... $(5^n)^{\frac{1}{5^K}}$

Assume

$(5^n)^{\frac{1}{5^K}} = 5$

$\frac{1}{5^K} \log_5 5^n = \log_5 5$

$\log_5 5^n = 5^K$

$n \log_5 5 = 5^K$

$5^K = n$

$\log_5 5^K = \log n$

$K = \log_5 n$

$\downarrow$

$\Theta(\log n)$.

15) for (i = 2 ; i <= 2^n ; i = i^2)

i = 2 $\quad 2^2 \quad 2^{2^2} \quad 2 \quad \ldots \quad 2^{2^K}$

$2^{2^K} = 2^n$

$\log_2 2^{2^K} = \log_2 2^n$

$2^K \log_2 2 = n \log_2 2$

$2^K = n$

$K \log_2 2 = \log_2 n$

$K = \log_2 n$

$\downarrow$

$\Theta(\log n)$

16) for (i = 8 ; i <= n ; i = i3)
{ for (j = n ; j = ~~#~~1 ; j = j/2)
{
for(K = 1 ; K <= n ; K = K * 4)
}
}

$\rightarrow$

for (K = 1 ; K <= n ; K = K * 4)

i = 1 $\quad 1 \times 4 \;,\; 4 \times 1 \times 4^2 \;,\; 16 \times 1 \times 4 \;\; 4^3 \ldots \quad K \times 1 \times 4 \;\; 4^K$

Assume

$K \times 1 \times 4 = n$

$K = \frac{n}{4 \times 1}$ $\rightarrow \Theta(n)$

$4^K = n \rightarrow K = \log_4(n)$

For (j = n ; j >= 1 ; j = j / 2)

$i = n \quad \frac{n}{2} \quad \frac{n}{2^2} \quad \cdots \quad \frac{n}{2^k}$

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

$$2^k = n$$

$$k \log_2 2 = \log_2 n$$

$$k = \log n \longrightarrow \Theta(\log n)$$

For (i = 3 ; i <= n ; i = i^3)

$i = 3, \quad 3^3, \quad 3^{3^2}, \quad 3^{3^3} \quad \cdots \quad 3^{3^k}$

Assume $3^{3^k} = n$

$$3^k \log_3 3 = \log_3 n$$

$$k \log_3 3 = \log_3 \log_3 n$$

$$k = \log_3 \log_3 n$$

$$\downarrow$$

$$\Theta(\log(\log n))$$

$\log(n) \cdot \log(n) \cdot (\log(\log n))$

$$\downarrow$$

$$\log^2(n) \cdot \log \log n$$

17) For (i = 1; i <= n; i++) $\longrightarrow n+1$

For (j = 1 ; j <= i; j++) $\longrightarrow n$

$\downarrow$

i = 1	i = 2	i = 3	...	i = n
j = 1	j = 1	j = 1		j = 1
	j = 2	j = 2		⋮
		j = 3		n.

$(n+1) \times (n)$

$O(n^2)$

18) For(i=1; i<=n; i++) → n+1.
For(j=1; j<=n; j=j+i)
a=a+5;

→

i = 1	i = 2	i = 3	i = n
For(j=1; j<=n; j=j+1)	j+2	j+3	
n	$\frac{n}{2}$	$\frac{n}{3}$	$\frac{n}{n}$

$\Rightarrow n + \frac{n}{2} + \frac{n}{3} \cdots\cdots + \frac{n}{n}$

$n\left(1 + \frac{1}{2} + \frac{1}{3} \cdots \frac{1}{n}\right) = n\log n \quad \Theta(n\log n)$

* Recurrance Relation.
recurrance relation is mathematical expression that describe overall cost of problem in term of cost of solving smaller sub-problem.

Algo Fact(n) → $T(n) \rightarrow \begin{cases} n * T(n-1) & n>1 \\ 1 & n=1 \end{cases}$

if (n == 1)
return 1. → 1 unit
else
return n * Fact(n-1) → n × T(n-1)

Substitution method.

Example

1) $T(n) = \begin{cases} T(n-1) + c & \text{if } n > 1 \\ c & n = 1 \end{cases}$

$T(n) = T(n-1) + c$ — (1)

put n as n-1

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

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

put value of $T(n-1)$ in (1)

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

$T(n) = T(n-2) + 2c$ — (2)

put n as $n-2$ in eqn 1

$T(n-2) = T(n-3) + c$

put $T(n-2)$ in (2)

$T(n) = T(n-3) + 3c$ — (3)

!

$T(n) = T(n-k) + kc$ — (4)

Assume $n-k = 1$

$k = n-1$

put value of k in eqn (4)

$T(n) = T(n-[n-1]) + (n-1)c$

$= T(1) + nc - c$

$= c + nc - c$

$= nc$

$= \Theta(n)$

2) $T(n) = \begin{cases} T(n-1) + n & n > 1 \\ 1 & n = 1 \end{cases}$

→ $T(n) = T(n-1) + n$ ⟶ (1)

put n as $n-1$

$T(n-1) = T(n-2) + (n-1)$ —

put value of $T(n-1)$ in (1)

↓

$T(n) = T(n-2) + (n-1) + n$ ⟶ (2)

$= T(n-2) + 2n - 1$ ⟶ (3)

$1+2+3+\cdots+(n-1)+n \rightarrow \frac{n(n+1)}{2}$

put n as n-2

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

$= T(n-3) + (n-2)$

put in (2)

$T(n) = T(n-3) + (n-2) + 2n - 1 + (n-1) + n$

$= T(n-3) + 3n - 3$

$T(n) = T(n-3) + (n-2) + (n-1) + n$

$\vdots$

$T(n) = T(n-K) + (n-(K-1)) + (n \cdots (n-1) + n$ (4)

Assume $n-K=1$

$K = n-1$ AP

$T(n) = T(n-(n-1)) + (n-(n-1)-1) + \cdots (n-1)+n$

$n-(n-2)$

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

$= 1+2 \cdots + (n-1) + n$

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

$= 1+\frac{n^2+n}{2}$

$= 1+n^2$

$= O(n^2)$

3) $T(n) = \begin{cases} T(n-1) + \log n & n > 1 \\ 1 & n = 1 \end{cases}$

In this type of example the complexity is ~~div~~ n multiply second ter

$T(n-1) + c \rightarrow n$

$+n \rightarrow n^2$

$+\log n \rightarrow n \log n$

→

$T(n) = T(n-1) + \log n$ —— (1)

put n as n-1

$T(n-1) = T(n-1-1) + \log(n-1)$

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

put $T(n-1)$ in (1)

$T(n) = T(n-2) + \log(n-1) + \log(n)$ —— (2)

put n as n-2 in (1)

$T(n-2) = T(n-3) + \log(n-2)$

put $T(n-2)$ in (2)

$T(n) = T(n-3) + \log(n-2) + \log(n-1) + \log(n)$

⋮

$T(n) = T(n-K) + \log(n-(K-1)) + \dots \log(n-1) + \log(n)$

Assume $n-K = 1$

$K = n-1$

$T(n) = T(n-(n-1)) + \log(n-(n-2)) + \dots \log(n-1) + \log n$

$= T(1) + \log$

$= 1 + \log(2) + \dots \log(n-1) + \log n$

$= 1 + \log(n!)$

$T(n) = 1 + n\log n$ → $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$

↓

$O(n\log n)$

$\log n! = \log\sqrt{2\pi n} + \log\left(\frac{n}{e}\right)$

↓

$\frac{1}{2}\log(2\pi n) + n\log n - n$

ignoring constant term

$\log(n!) \leftrightarrows n\log n$

4) $T(n) = \begin{cases} T(n-1) * n & n > 1 \\ 1 & n = 1 \end{cases}$

→

$T(n) = T(n-1) * n$ ——— (1)

$T(n-1) = T(n-1-1) * (n-1)$

Put $T(n-1)$ in (1)

$T(n) = T(n-2) * (n-1) * n$ —(2)

$T(n) = T(n-3) * (n-2) * (n-1) * n$

⋮

$T(n) = T(n-k) * (n-(k-1)) * \cdots n$

Assume $n - k = 1$

$k = n-1$

$T(n) = T(n-(n-1)) * (n-(n-1)) * (n-(n-2)) * \cdots$

Factorial

$= T(1) * 1 * 2 * \cdots n$

$= 1 * 2 * \cdots n$

$T(n) = n!$

↓

$O(n!)$

5) $T(n) = \begin{cases} 2T(n-1) + n & n > 1 \\ 1 & n = 1 \end{cases}$

→

$T(n) = 2T(n-1) + n$ ——— (1)

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

$T(n) = 2 * [2T(n-2) + (n-1)] + n$

$T(n) = 2^2 T(n-2) + 2(n-1)] + n$ —(2)

Put n as $n-2$

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

$T(n) = 2^2 [2T(n-3 + (n-2)] + 2(n-1) + n$

GP = $\frac{r(r^n-1)}{r-1}$ $\frac{A(r^n-1)}{r-1}$ Increasing order.

$T(n) = 2^3 T(n-3) + 2^2(n-2) + 2(n-1) + n$

$\vdots$

$T(n) = 2^K T(n-K) + 2^{K-1}(n-(K-1) + 2^{K-2}(n-(K-2) \cdots + n$

Assume $n-K = 1$

$K = n-1.$

$T(n) = 2^{n-1} T(n-(n-1) + 2^{n-2}(n-(n-1)-1) +$
$2^{n-3}(n-(n-1)-2) + \cdots n$

$= 2^{n-1} + 2^{n-2}(2) + 2^{n-3}.3 \cdots 2^2(n-2) +$
$2(n-1) + n$

→ Reverse

$= 2^0(n) + 2^1(n-1) + 2^2(n-2) + \cdots$
$2^{(n-2)}(2) + 2^{n-1}(1)$

multiply 2 to both side.

$2T(n) = 2^1(n) + 2^2(n-1) + 2^3(n-2) + \cdots$
$2^{n-1}(2) + 2^n \cdot 1$

Substract 4 from 5

$2T(n) - T(n) = -n + 2^1 + 2^2 + 2^3 \cdots\cdots 2^{n-1} + 2^n$

Geometric progression.

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

$= -n + 2^{n+1} - 2$

$= 2^n 2 - n - 2$

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

↓

$O(2^n)$

5) $T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & n > 1 \\ 1 & n = 1 \end{cases}$

→

$T(n) = 2T\left(\frac{n}{2}\right) + n$ —— (1)

Put n as $n-1$

$T(n-1) = 2T\left(\frac{n-1}{2}\right) + (n-1)$

$T(n) = 2$

n as $\frac{n}{2}$

$T\left(\frac{n}{2}\right) = 2T\left(\frac{n}{4}\right) + \frac{n}{2}$

$T(n) = 2\left[2T\left(\frac{n}{4}\right) + \frac{n}{2}\right] + n$

$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + \frac{n}{2} \times 2 + n.$

$T(n) = 2^2 T\left(\frac{n}{2^2}\right) + 2n \quad (n+n)$ —— (2)

put $\left(\frac{n}{2^2}\right)$ in (1)

$T\left(\frac{n}{2^2}\right) + 2\left(T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)\right)$

put in (2)

$T(n) = 2^2\left[2T\left(\frac{n}{2^3}\right) + \left(\frac{n}{2^2}\right)\right] + 2n$

$T(n) = 2^3 T\left(\frac{n}{2^3}\right) + 3n$ —— (3)

⋮

$T(n) = 2^K T\left(\frac{n}{2^K}\right) + Kn$ —— (4)

Assume $\frac{n}{2^K} = 1$

$T(n) = \sqrt{n} * T(\sqrt{n}) + n$

$T(n^{1/2}) = n^{1/4} * T(n^{1/4}) + n^{1/2}$

$\sqrt{\sqrt{n}} = n^{1/4}$

$T(n) = n^{3/4} + n^{1/2} * (n^{1/4} + T(n^{1/4}) + n^{1/2} + n)$

$n = 2^k$

$\log_2 n = k \log_2 2$

$k = \log n$

$\downarrow$

~~$O(\log n)$~~

$T(n) = 2^k T(1) + n \log n$

$= n + n \log n \qquad (2^k = n)$

$\downarrow$

$O(n \log n)$ as dominating terom is $n \log n$.

6) $T(n) = \begin{cases} \sqrt{n} \cdot T(\sqrt{n}) + n & n > 2 \\ 2 & n = 2 \end{cases}$ $n \log \log n$

$\rightarrow$

$T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$ ——— (1)

put n as $\sqrt{n}$ in (1)

~~$T(\sqrt{n}) =$~~

$T(n^{1/2}) = n^{1/4} \cdot T(n^{1/4}) + \sqrt{n}$

$T(n) = \sqrt{n} * [n^{1/4} \cdot T(n^{1/4}) + \sqrt{n}] + n$

$= n^{1/2} [n^{1/2^2} T(n^{1/2^2}) + n^{1/2}] + n$

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

~~$T(\sqrt{n}) =$~~

$T(n) = \sqrt{n}\, T(\sqrt{n}) + n$ ——— (1)

n as $\sqrt{n}$

$T(\sqrt{n}) = n^{1/2^2} T(n^{\frac{1}{2^2}}) + n^{\frac{1}{2}}$

put in 1

$T(n) = \sqrt{n} \left(n^{\frac{1}{2^2}} T(n^{\frac{1}{2^2}}) + n^{\frac{1}{2}} \right) + n$

$n^{3/4} + (n^{1/4}) + 2n$

$n^{\frac{2^K - 1}{2^K}} + (n^{\frac{1}{2^K}}) + kn$

$\frac{1}{2^K} = n \rightarrow K = \log n$

$n^{\frac{2^K - 1}{2^K}} T(1) + n \log n$

$2^{\log n} - 1 \quad 1 - \frac{1}{\log n}$

$T(n) = 2T\sqrt{n} + 1 \quad n > 1$

$2 \quad n = 2$

$$T(n) = \sqrt{n} \cdot n^{\frac{1}{2^2}} T(n^{\frac{1}{2^2}}) + n + n$$

($\sqrt{n} \cdot \sqrt{n}$)

$$= n^{\frac{1}{2^3}} T(n^{\frac{1}{2^2}}) + 2n \quad \text{—— (2)}$$

$$= \sqrt[8]{n} \cdot T(\sqrt[4]{n}) + 2n$$

put n as $n^{\frac{1}{4}}$

$$T(n^{\frac{1}{4}}) = n^{\frac{1}{2^3}} T(n^{\frac{1}{2^3}}) + n^{\frac{1}{2^2}}$$

$$T(n) = n^{\frac{1}{2^3}} [n^{\frac{1}{2^3}} \cdot T(n^{\frac{1}{2^3}}) + n^{\frac{1}{2^2}}] + 2n$$

$$T(n) = \sqrt[16]{n} \cdot T(\sqrt[8]{n}) + \sqrt[8]{n} \sqrt[4]{n} + 2n$$

Ans :→

$$T(n) = \sqrt{n}\, T(\sqrt{n}) + n$$

$$T(n) = n^{1/2}\, T(n^{1/2}) + n \quad \text{—— (1)}$$

n as $n^{1/2}$

$$T(n^{1/2}) = (n^{1/4})\, T(n^{1/4}) + n^{1/2} \quad \text{—— (2)}$$

$$T(n) = n^{1/2} [(n)^{1/4}\, T(n^{1/4}) + n^{1/2}] + n$$

$$= n^{3/4}\, T(n^{1/4}) + n + n$$

$$T(n) = n^{3/4}\, T(n^{1/4}) + 2n \quad \text{—— (2)}$$

$$T(n) = n^{3/4} [n^{1/8}\, T(n^{1/8}) + n^{1/4}] + 2n$$

$$= n^{7/8}\, T(n^{1/8}) + 3n \quad \text{—— (3)}$$

$$T(n) = n^{\frac{2^k - 1}{2^k}}\, T(n^{\frac{1}{2^k}}) + kn \quad \text{—— (4)}$$

Assume. $\frac{1}{2^K} \log n \quad n^{1/2^K} = 2$

$\log_2$ on b.s.

$$\frac{1}{2^K} \log n = \log_2 2$$

$$2^K = \log n$$

Apply log.

$$K = \log\log n$$

$$T(n) = n^{\frac{\log n - 1}{\log n}} \cdot 2 + n \log\log n$$

$$= n^{1 - \frac{1}{\log n}} \cdot 2 + n \log\log n$$

$$= \frac{n}{n^{1/\log n}} \cdot 2 + n \log\log n$$

$$= \frac{n}{2} \cdot 2 + n \log\log n$$

$$= n + n \log\log n$$

$$\downarrow$$

$$O(n \log\log n)$$

✓7) $T(n) = \begin{cases} 2T(\sqrt{n}) + 1 & n > 2 \\ 2 & n = 2 \end{cases}$

→

$$T(n) = 2\,T(n^{1/2}) + 1$$

n as n/2

$$T(n/2) = 2\,T(n^{1/4}) + 1$$

$$T(n) = 2\,[2 \cdot T(n^{1/4}) + 1] + 1$$

$$T(n) = 2^2\,T(n^{1/4}) + 2$$

n as n/4

$T(n/4) = 2 \cdot T(n^{1/8}) + 1$

$T(n) = 2^2 [2 \cdot T(n^{1/8}) + 1] + 2$

$= 2^3 T(n^{1/2^3}) + 3$

!

$T(n) = 2^K \cdot T(n^{\frac{1}{2^K}}) + K.$

Assume $n^{\frac{1}{2^K}} = 2$

$\frac{1}{2^K} \log n = \log_2 2$

$2^K = \log n$

$K = \log\log n$

$T(n) = \log n \cdot 2 + \log\log n$

$T(n) = O(\log n)$ as $\log n$ is dominating.

8) $T(n) = \begin{cases} \sqrt{2}\, T(\frac{n}{2}) + \sqrt{n} & n > 1 \\ 1 & n = 1 \end{cases}$

→

$T(n) = \sqrt{2}\, T(n/2) + \sqrt{n}$

n as n/2

$T(n/2) = \sqrt{2}\, T(n/4) + (n/2)^{\frac{1}{2}}$

$T(n) = \sqrt{2} [\sqrt{2} T(n/4) + (n/2)^{\frac{1}{2}}] + (n)^{\frac{1}{2}}$

$= 2\, T(n/4) + \sqrt{2} (n/2)^{\frac{1}{2}} + n^{\frac{1}{2}}$

$\sqrt{2}$ $\frac{\sqrt{n}}{\sqrt{2}}$ $2\sqrt{n}$

$a^{\log_2 b} \rightarrow b^{\log_2 a}$

n as n/4

$$T(n/4) = \sqrt{2}\, T(n/8) + (n/4)^{\frac{1}{2}}$$

$$T(n) = (\sqrt{2})^2 \left[\sqrt{2}\, T(n/8) + (n/4)^{\frac{1}{2}} \right] + \frac{\sqrt{2}\,(n/2)^{1/2}}{2\sqrt{n}} *$$

$$= (\sqrt{2})^3 T\left(\frac{n}{2^3}\right) + (\sqrt{2})^2 \frac{\sqrt{n}}{(\sqrt{2})^2} + 2\sqrt{n}$$

$$= (\sqrt{2})^3 T\left(\frac{n}{2^3}\right) + 3\sqrt{n}$$

$$T(n) = (\sqrt{2})^k T\left(\frac{n}{2^k}\right) + k\sqrt{n}$$

Assume $\frac{n}{2^k} = 1$

$$2^k = n$$

log on b.s

$$k = \log n$$

$$T(n) = (\sqrt{2})^{\log n} T(1) + \sqrt{n} \log n$$

$$= (2^{1/2})^{\log n} + \sqrt{n} \log n$$

$$= \Theta(\sqrt{n}$$

$$= (n)^{\log_2 \sqrt{2}} + \sqrt{n} \log n$$

$$= n^{\log_2 (2)^{1/2}} + \sqrt{n} \log n$$

$$= \sqrt{n} + \sqrt{n} \log n$$

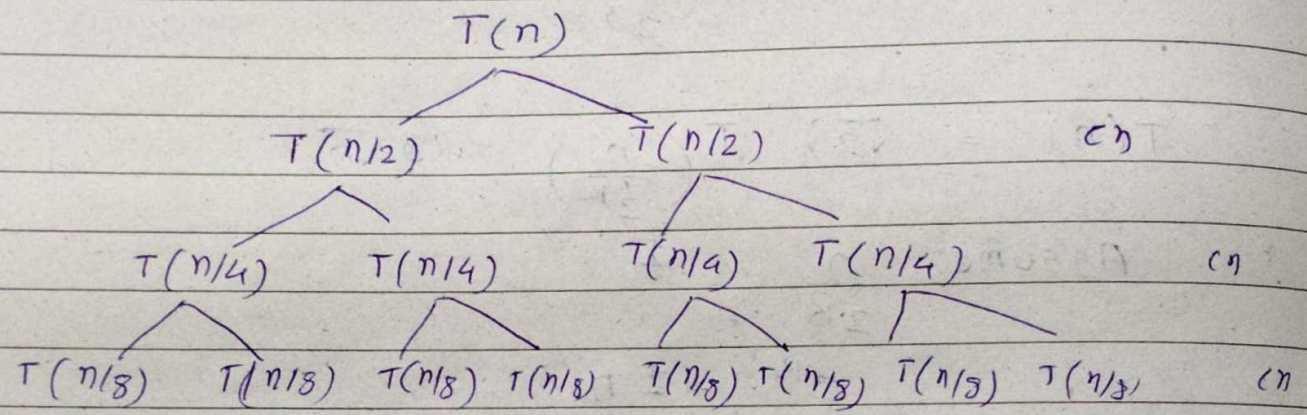
$$\downarrow$$

$$O(\sqrt{n} \log n)$$

* Recurrance Tree method.

1) $T(n) = \begin{cases} 2T(n/2) + cn & n > 1 \\ 1 & n = 1 \end{cases}$

→ $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right) + cn$



$\frac{n}{2^k} = 1.$

$2^k = n$

$k = \log n$

$T(n) = T\left(\frac{n}{2^k}\right) + cn + cn + cn - - - k$

$= T\left(\frac{n}{2^k}\right) + kcn$

$= 1 + \log n \cdot c \cdot n$

$= 1 + n\log n.$

$\downarrow$

$O(n \log n).$

* Master Theorem

1) $T(n) = a\,T(n/b) + \Theta(n^k \log^p n)$

$a \geq 1$, $b > 1$, $k \geq 0$ & p is real no.

case 1: if $a > b^k$ then $T(n) = \Theta(n^{\log_b a})$

case 2: if $a = b^k$ then

1) $p > -1$ then $T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$
2) $p = -1$ then $T(n) = \Theta(n^{\log_b a} \log \log n)$
3) $p < -1$ then $T(n) = \Theta(n^{\log_b a})$

case 3: if $a < b^k$ then

1) $p \geq 0$ then $T(n) = \Theta(n^k \log^p n)$
2) $p < 0$ then $T(n) = \Theta(n^k)$

1) $T(n) = 4T(n/2) + n$

$a = 4$ $b = 2$ $k = 1$ $p = 0$

a b^k

$4 > 2$

then $T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^{\log_2 4})$

$= \Theta(n^2)$

2) $T(n) = 2T(n/2) + n$

$a = 2$ $b = 2$ $k = 1$ $p = 0$

a b^k

$2 = 2$

i) $p > -1$

$\therefore T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$= \Theta(n \log n)$

$= \Theta(n \log n)$

3) $T(n) = 4T(\frac{n}{2}) + n^2$

→ $a=4 \quad b=2 \quad K=2 \quad P=0$

a	b^K
4	2^2
4	= 4

i): $T(n) = \Theta(n^{\log_b a} \cdot \log^{P+1} n)$

$= \Theta(n^{\log 4} \log n)$

$= \Theta(n^{\log_2 4} \log n)$

$= \Theta(n^2 \log n)$

4) $T(n) = 8T(n/2) + n^2$

$a=8 \quad b=2 \quad K=2$

$a > b^K$

$8 > 4$

$T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^{\log_2 8})$

$T(n) = \Theta(n^3)$

5) $T(n) = 4T(n/2) + n \log n)$

$a=4 \quad b=2 \quad K=1 \quad P=1$

$a > b^K$

$4 > 2$

$T(n) = \Theta(n^{\log_b a})$

$= \Theta(n^{\log_2 4})$

$= \Theta(n^2)$

6) $T(n) = 4T(n/2) + n^2 \log n$

$a=4 \quad b=2 \quad K=2 \quad P=1$

$4 = 2^2$

$4 = 4$

i) $T(n) = \Theta(n^{\log_b a} \log^{P+1} n)$

$= \Theta(n^2 \log^2 n)$

7) ~~T(n)~~ =

$T(n) = 4T(n/2) + n \rightarrow n^2$

$T(n) = 2T(n/2) + 1 \rightarrow n$

$T(n) = 2T(n/2) + \log n \rightarrow n \log n$

$T(n) = 2T(n/2) + n \rightarrow n \log n$

7) $T(n) = 2T(n/2) + n/\log n$

$a = 2 \quad b = 2 \quad K = 1 \quad P = -1$

$a = b$

$2 = 2$

iii) $P = -1$

$T(n) = (n^{\log_b a} \log \log n)$

$= \Theta(n \log \log n)$

8) $T(n) = 2T(n/2) + n/\log^2 n$

$\Theta(n)$. (2 - iii)

9) $T(n) = 2T(n/2) + n^2$

$a = 2 \quad b = 2 \quad K = 2 \quad P = 0$

$a < b^K$

$2 < 4$

$T(n) = \Theta(n^K \log^P n)$

$= \Theta(n^2 \log^0 n)$ $\log^0 n = 1$.

$= \Theta(n^2)$

10) $T(n) = 2T(n/2) + n/\log^4 n$

→ $a = 2 \quad b = 2 \quad K = 1 \quad P = 4$

$2 = 2$

$T(n) = $ ~~$n(\log$~~

$= \Theta$

11) $T(n) = 3T(n/2) + \frac{n^2}{\log n}$

→ $a = 3 \quad b = 2 \quad K = 2 \quad P = -1$

a

$3 < 4$

$T(n) = \Theta(n^K) \rightarrow \Theta(n^2)$

12) $T(n) = \sqrt{2}\, T(n/2) + \log n$

$a = \sqrt{2} \quad b = 2 \quad K = 0 \quad P = 1$

$\sqrt{2} > 1$

$T(n) = \Theta(n^{\log_2 \sqrt{2}})$

$T(n) = \Theta(\sqrt{n})$

13) $T(n) = T(\sqrt{n}) + \log n$

→ $a = 1 \quad b = 1 \quad K = 0 \quad P = 1$

$a = b^K$

$1 = 1$

$T(n) = \Theta(n \log^2 n)$.

$T(n) = T(n^{1/2}) + \log n$

Assume $n = 2^m \rightarrow m = \log n$

$(2^m)^{1/2} = 2^{m/2}$

$T(2^m) = T(2^{m/2}) + \log 2^m$.

$= T(2^{m/2}) + m \qquad \log_2 2$

$T(2^m) = S(m) \qquad \therefore T(2^{m/2}) = S(m/2)$.

$S(m) = S(\frac{m}{2}) + m$

$a = 1 \quad b = 2 \quad K = 1 \quad P = 0$

$a < b$

$n^K \log^{P+1} n$

$m \log^0 n \rightarrow m = \log n$

$\log_2 n = \log_{2^2} m$ $\quad m = \log n$.

$1 < 2$.

i) $P \geq 0$

$T(n) = \Theta(n^k \log^p n)$

$= \Theta(n)$ $\quad \Theta(m' \log n)$.

$\Theta(m) \rightarrow \Theta(\log n)$

14) $T(n) = 2T(\sqrt{n}) + \log n$.

→ $T(n) = 2T(n^{1/2}) + \log n$.

Assume $n = 2^m$

Apply log

$m = \log n$

$T(2^m) = 2T(2^{m/2}) + m$

Assume $T(2^m) = S(m)$

$S(m) = 2S\left(\frac{m}{2}\right) + m$.

Apply master thm

$a = 2 \quad b = 2 \quad k = 1 \quad P = 0$

$a = b^k$

$2 = 2$

$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$T(n) = \Theta(m^{\log_2 2} \log m)$

$= \Theta(m \log m)$

$= \Theta(\log n \log\log n)$.

$S(m) = \frac{m}{2} 2^{m/2} S\left(\frac{m}{2}\right) + n \cdot 2^m$

$a = 2^m \quad b = 2 \quad k = 1 \quad P =$

15) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

→ $T(n) = n^{1/2} T(n^{1/2}) + n$

$n = 2^m$

$m = \log n$.

$T(2^m) = 2^{m/2} T(2^{m/2}) + 2^m$

Assume $T(2^m) = S(m)$

$S(m) =$

Divide 2^m to both side.

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2}) \cdot 2^{m/2}}{2^m} + 1.$$

$$= \frac{2^{m/2} T(2^{m/2})}{2^{m/2}\, 2^{m/2}} + 1$$

$$\frac{T(2^m)}{2^m} = \frac{T(2^{m/2})}{2^{m/2}} + 1$$

Assume $\frac{T(2^m)}{2^m} = S(m)$

$S(m) = S(m/2) + 1$

Apply master th

$a = 1,\ b = 2,\ K = 0,\ P = 0$

$a = b^K$

$1 = 2^0$

$T(n) = \Theta(n^{\log_b a} \log^{P+1} n)$

$S(m) = \Theta(m^{\log_2 1} \log m)$

$= \Theta(\log m)$

$\frac{T(2^m)}{2^m} = \Theta(\log \log n)$

$T(2^m) = 2^m \log \log n$

$T(n)$

$T(n) = \Theta(n \log \log n)$

16) $T(n) = T(\sqrt{n}) + 1$

→ $T(n) = T(n^{1/2}) + 1$.

$n = 2^m$

$m = \log n$.

$T(2^m) = T(2^{m/2}) + 1$

$T(2^m) = S(m)$

$S(m) = S(\frac{m}{2}) + 1$.

$a = 1, \quad b = 2, \quad k = 0, \quad p = 0$.

$a \quad b^k$

$1 = 2^0$.

$T(n) = \Theta(n^{\log_b a} \log^{p+1} n)$

$S(m) = \Theta(m^{\log_2 1} \log m)$

$= \Theta(\log m)$

$T(2^m) = \Theta(\log \log n)$

- Greedy Method.
  - does not give optimal soln all time.

* Fractional Knapsack

Q. Find optimal soln for knapsack instance $n = 7$ $m = 15$. Profit 10, 5, 15, 7, 6, 18, 3 and their weight 2, 3, 5, 7, 1, 4, 1.

→

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
P_i/ω_i	$\frac{10}{2}$	$\frac{5}{3}$	$\frac{15}{5}$	$\frac{7}{7}$	$\frac{6}{1}$	$\frac{18}{4}$	$\frac{3}{1}$
=	5	1.66	3	1	6	4.5	3

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
P	10	5	15	7	6	18	3
ω	2	3	5	7	1	4	1

Fractional Knapsack → complexity $O(n)\log n)$

Arrange in decreasing order

P_i/ω_i = 6, 5, 4.5, 3, 3, 1.66, 1

i_5 i_1 i_6 i_3 i_4 i_2 i_4

items(x_i)	$\omega_i x_i$	
$\{i_1, i_2, i_3, i_4, i_5, i_6, i_7\}$	15 - 1	6 x 1 = 6
{0,0,0,0,1,0,0}	= 14	
{1,0,0,0,1,0,0}	14 - 2 = 12	6 + 10 = 16
{1,0,0,0,1,1,0}	12 - 4 = 8	16 + 18 = 34
{1,0,1,0,1,1,0}	8 - 5 = 3	34 + 15 = 49
{1,0,1,0,1,1,1}	3 - 1 = 2	49 + 3 = 52
{1,1,1,0,1,1,1}	$2 - \frac{2}{3} = 0$	$52 + \frac{2}{3} \times 5$
		= 56.3

Q.2) let's a, b, c, d, e, F, g be a collection of object with profit-weight values are follows.

a = 12, 4 c = 8, 5 e = 14, 3 g = 9, 6

b = 10, 6 d = 11, 7 F = 7, 1

What is optimal soln to fractional knapsack problem. Assume capacity has weight = 18 what is complexity of this method?

→

	a	b	c	d	e	F	g
profit	12	10	8	11	14	7	9
weight	4	6	5	7	3	1	6
$P_i \omega_i$	3	1.66	1.6	1.57	4.66	7	1.5
	7	4.66	3	1.66	1.6	1.57	1.5

items x_i	$w_i x_i$	profit.
{a, b, c, d, e, f, g}	18 - 1 = 17	7
{a, b, c, d, e, 1, g}		
{a, b, c, d, e, 1, g}	17 - 3 = 14	7 + 14 = 21
{1, b, c, d, 1, 1, g}	14 - 4 = 10	21 + 12 = 33
{1, 1, c, d, 1, 1, g}	10 - 6 = 4	33 + 10 = 43
{1, 1, 1, d, 1, 1, g}	4 - 4 = 0	$43 + \frac{4}{5} \times 8 =$
		49.4

* Job sequencing problem Aim → TO find max. profit.

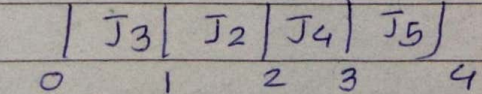
Jobs	J_1	J_2	J_3	J_4	J_5
Deadline	2	2	1	3	4
profits	20	60	40	100	80

max. deadline = 4.

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
profit	100	80	60	40	20

~~Gantt chart~~. ~~Giant chart~~.



→ Gantt chart.

Job Sequence = {J3, J2, J4, J5}

Profit = 40 + 60 + 100 + 80 = 280.

Q. given the Job, their deadline & profit shown

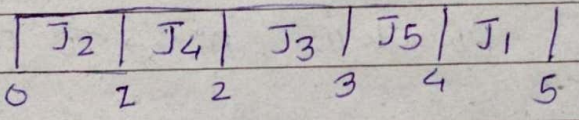
Job	J_1	J_2	J_3	J_4	J_5	J_6
Deadline	5	3	3	2	4	2
profit	200	180	190	300	120	100

write optimal schedule that give max profit (ii) are all job completed in optimal schedule (iii) what is max earn profit

→

i) max. deadline = 5

J	J_4	J_1	J_3	J_2	J_5	J_6
D	2	5	3	3	4	2
P	300	200	190	180	120	100



→ Gantt chart

Job seq = $\{J_2, J_4, J_3, J_5, J_1\}$ → optimal schedule

profit → 180 + 300 + 190 + 120 + 200.

= 990

ii) No. not all job completed in optimal schedule.

iii) max earn profit = 990.

Q. $J_1, J_2 \dots J_9$

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

Find max. profit.

J	J_3	J_9	J_7	J_2	J_4	J_5	J_8	J_1	J_6
D	5	3	2	2	3	4	7	7	5
P	30	25	23	20	18	18	16	15	180

max. deadline = 7.

	J_2	J_7	J_9	J_5	J_3	J_1	J_8	→ Gantt chart.
	0	1	2	3	4	5	6	7

Job seq → $\{J_2, J_7, J_9, J_5, J_3, J_1, J_8\}$

profit → 5 + 20 + 23 + 25 + 18 + 30 + 15 + 16

⇒ 147

* Huffman coding.
  - use for encryption & decryption.
  - To compress data.
  - HC is lossless data compression algorithm the idea is to assign variable length code to i/p character. length of assign code are based on freq. of corresponding char.
  - the variable length code assign to i/p char are prefix code. means that code are assign in such way that code assign to one char. is not prefix of code assign to any other char. the code length of char depend on how frequently it occur in given text.
  - The char which occur most frequently get smallest code & the one who occur least frequently get largest code.
  - There are 2 major steps in HC.

1) Building huffman tree from i/p char.
2) assigning code to char by traversing Huffman tree.

- Steps involve to construct Huffman
1) Create link node for each char of the text. leaf node of char contain occuring freq. of that char.
2) arrange all the node in increasing order of their freq. value.
3) Consider 1st 2 node having minimum freq. create a new internal node. freq. of this new node is sum of freq of those 2 nodes. make 1st node as left child & other as right child of newly created node.
4) Keep repeating step 2 & 3 until all node form a single tree. The tree finally obtain is desired Huffman tree.

* Average code length per char = $\frac{\Sigma(\text{freq} \times \text{code length})}{\Sigma(\text{freq})}$

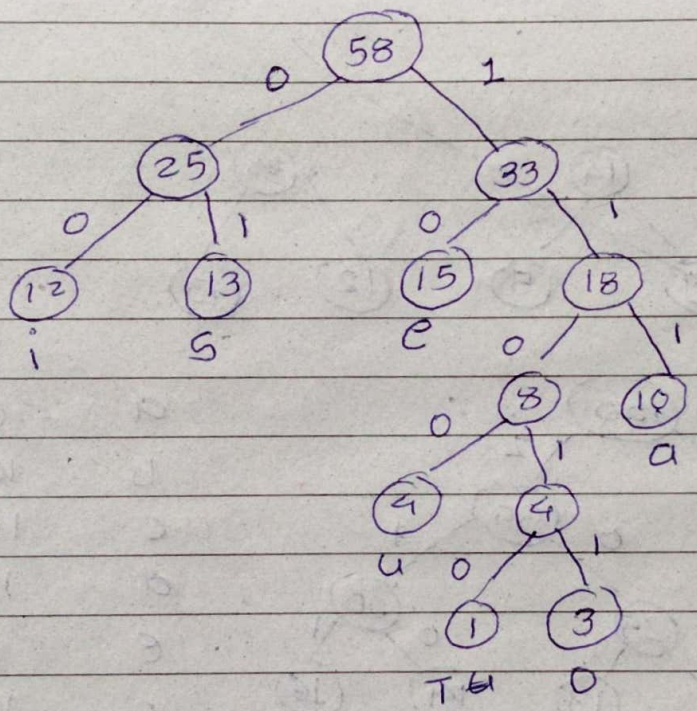
* Total no. of bit in huffman encodes msg
= total no. of ~~char in~~ msg. * frequency * avg code length per char.

Freqeuency → no. of occurence of char in data.

Q. File contain following character with freque as shown
if huffman coding used for data compression then determine 1) huffman code for each char. 2) avg code length 3) length of huffman endcoded msg in bit

a	e	i	o	u	s	t
10	15	12	3	4	13	1

→



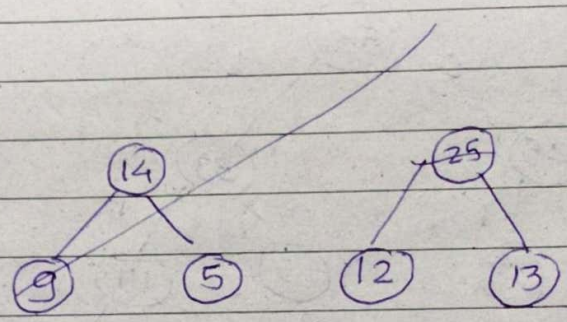
a	10	111	avg code = $\frac{\text{frequency} \times \text{codelength}}{\text{frequency}}$
e	15	10	
i	12	00	
o	3	11011	$= 10\times3 + 15\times2 + 12\times2 + 3\times5 + 4\times4 + 13\times2 + 1\times5$
u	4	1100	$\frac{}{58}$
s	13	01	
T	1	11010	
	58		$= 2.59$

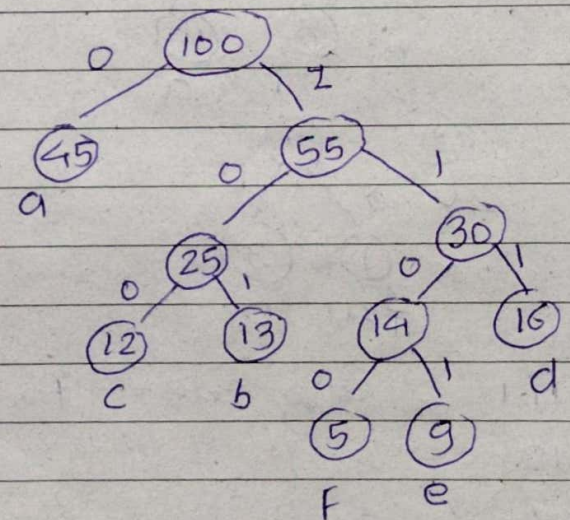
3) Total no. of bit = total no. of ~~char in msg~~
freq x avg code length
= 58 x 2.52
= 146 bits

Q.2) a b c d e f
45 13 12 16 9 5

Suppose we have 10^5 char in data file find out how many bit is require to send msg.

→





a	0
b	101
c	100
d	111
e	1101
F	1100

- avg code length = $\frac{45\times1 + 13\times3 + 12\times3 + 16\times3 + 9\times4 + 5\times4}{100}$

$= \frac{224}{100} = 2.24$

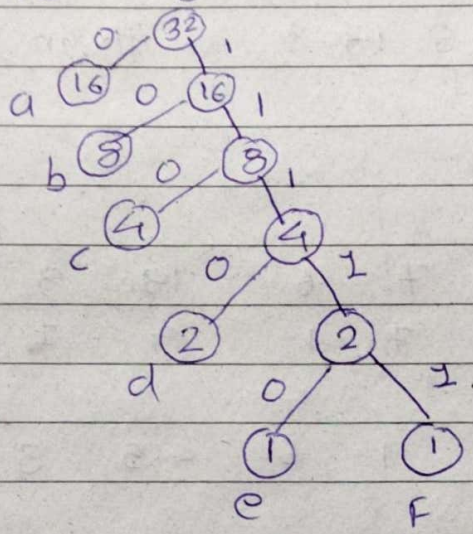
- total no. of bit = $2.24 \times 100 = 224$.

• total length = 2.2×10^5 bcoz we have
= 224000 bits. 10^5 char in data file.

Q.3

a	b	c	d	e	F	
$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$	$\times 32$
16	8	4	2	1	1	

→



a	16	0
b	8	10
c	4	110
d	2	1110
e	1	11110
F	1	11111

avg code = $\frac{16\times1 + 8\times2 + 4\times3 + 2\times4 + 1\times5 + 1\times5}{32}$

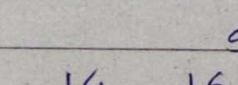
$= \frac{62}{32} = 1.93$

Total no of bit = $1.93 \times 32 = 62$

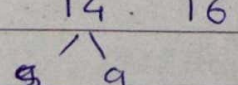
* 45 13 12 16 95 5 - take 2 min from list.

→ 5, 9, 12, 13, 16, 45 - add sum of this in list.

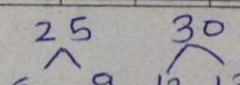
12, 13, 14, 16, 45



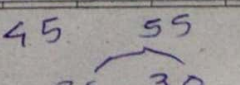
14 16 25 45



25 30 45



45 55



- Greedy method does not always gives optimal soln but some problem will solve only by greedy method like knapsack

* Fractional Knapsack problem.

Q. Find optimal soln for knapsack instance.
n = 7 (no. of items) m = 15 (knapsack capacity), profit
P_1 to P_7 10, 5, 15, 7, 6, 18, 3 & their wt are
2, 3, 5, 7, 1, 4, 1

→

m = 15.

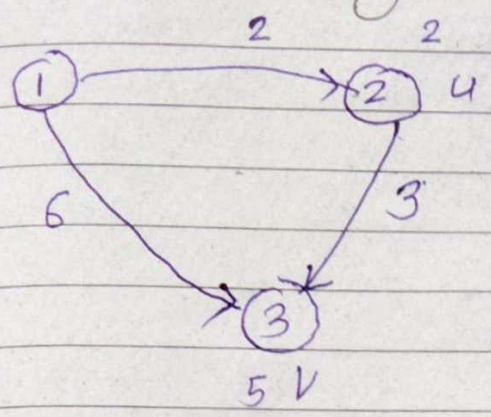
P	10	5	15	7	6	18	3
w	2	3	5	7	1	4	1
P/w	5	1.66	3	1	6	4.5	3

Arrange in decreasing order of their P/w ratio.

6, 5, 4.5, 3, 3, 1.66, 1
i_5 i_1 i_6 i_3 i_7 i_2 i_4.

→ already done.

* DijiKstra Algorithm → greedy Approach

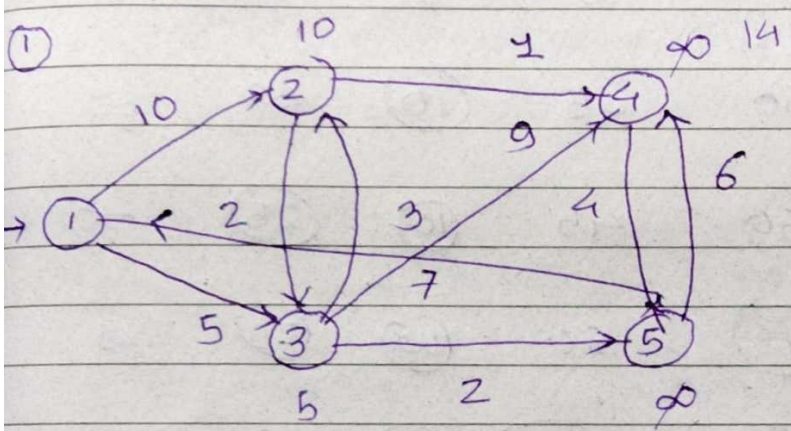


6 > 2 + 3

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

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

example :-



2 3 4 5

10 (5) ∞ ∞

↳ it is smallest

∴ we consider it

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

$\infty > 5 + 2$

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

update value of vertex 5 from ∞ to 5+2=7

2 → 8 (5+3)

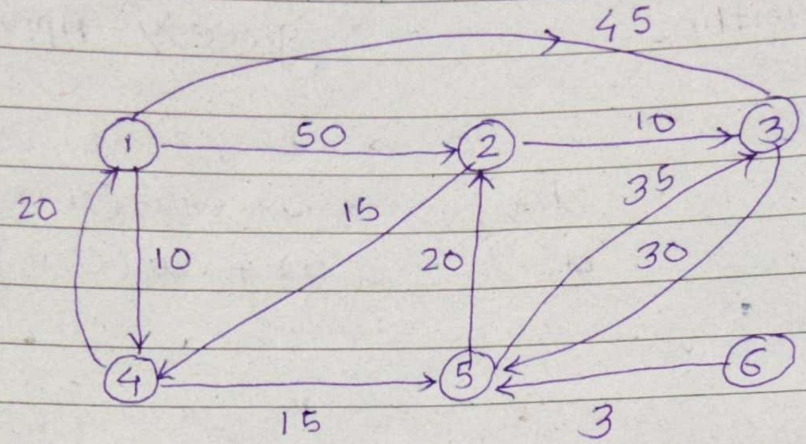
3 → 5 (5)

4 → 9 (5+3+1)

5 → 7 (5+2)

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)

2)



Vertex	Vertices	2	3	4	5	6
	choose with min wt	distance from 1				
4	{1, 4}	50	45	(10)	∞	∞
5	{1, 4, 5}	50	45	(10)	(25)	∞
2	{1, 4, 5, 2}	(45)	60	(10)	(25)	∞
3	{1, 4, 5, 2, 3}	(45)	(55)	(10)	(25)	∞
6	{1, 4, 5, 2, 3, 6}	(45)	(55)	(10)	(25)	(∞)

* Algorithm.

```
Dijikstra (G, ω, s)
Initialize Single Source (G, s)
  S ← φ
  Q ← v[G]                                  → O(V)
  while Q ≠ 0                               → O(V)
    u ← Extract min(Q)                      → V log V
    S ← S ∪ {a}                             → 1
  For each vertex v ∈ Adj[u]                → V² / E log
  do Relax (u, v, ω)          → 1              ↓        ↓
                                          Complete   den
                                          graph      gra
```

- Initialize Single source (G, s)
  For each vertex $v \in V[G]$
  do $d[v] \leftarrow \infty$
  $pred[v] = Nil$
  $d[s] \leftarrow 0$

- Relax (u, v, w)
  if $(d[v] > d[u] + w(u,v))$
  then
  $d[v] = d[u] + w(u,v)$
  $pred[v] \leftarrow u$.

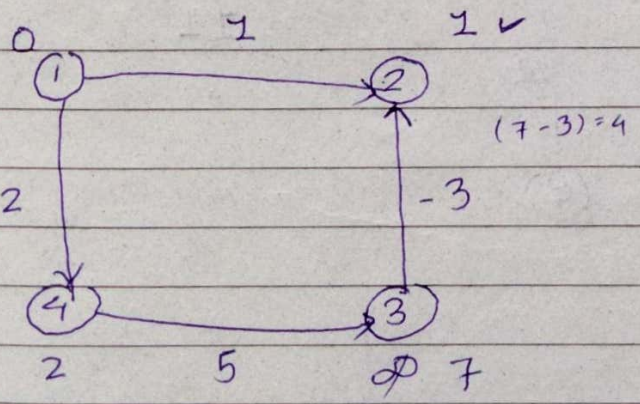
* Dijikstra :-

$$O(V) + O(V) + V\log V + 1 + E\log V$$

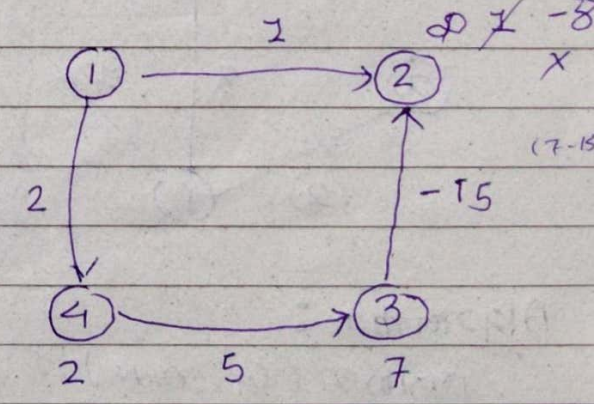
$$= O(E\log V) \;/\; O((V+E)\log V)$$

* limitation → if there is negative wt present in graph dijikstra doesn't work... .

ex :-



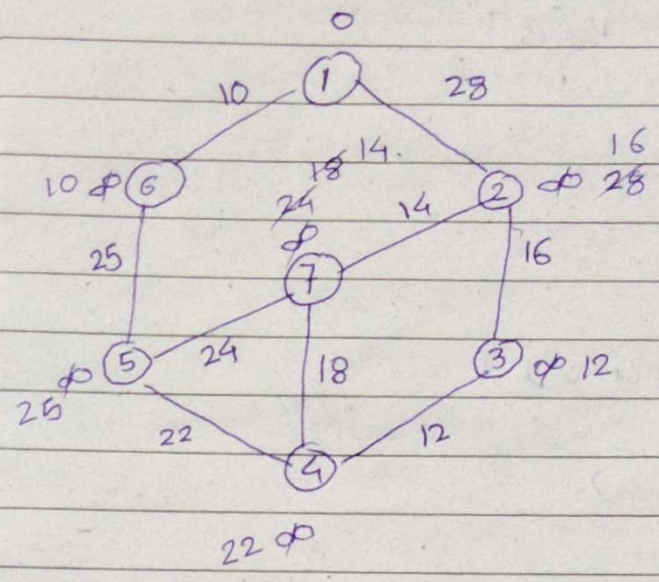
work.



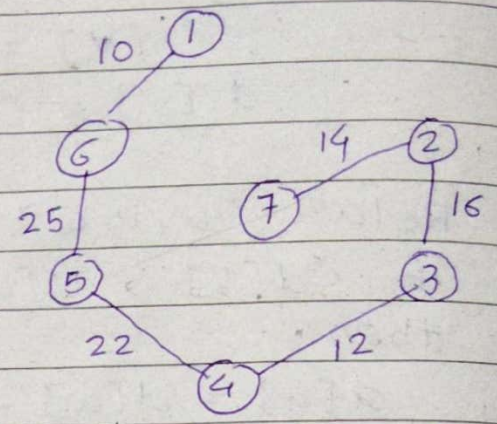
Not work as min become -8 & we already extract for this.

- this problem can be Solve using dynamic programming & other algorithm.

* prims algorithm



$\Rightarrow$

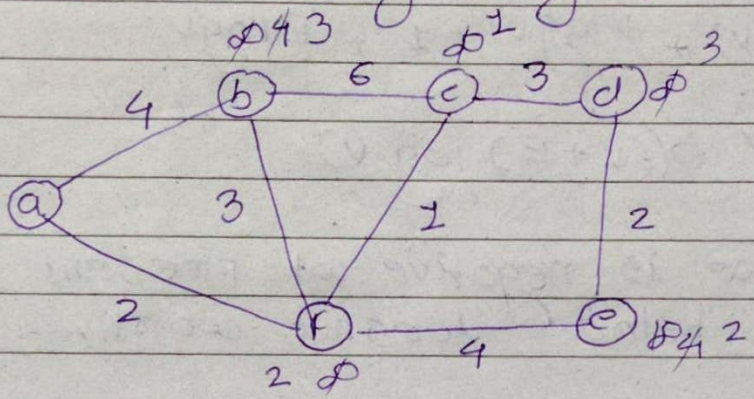


$\omega(u,v) > d[v]$

$d[v] = \omega(u,v)$

$= 99$ (total)

Q. Find MST of given graph.

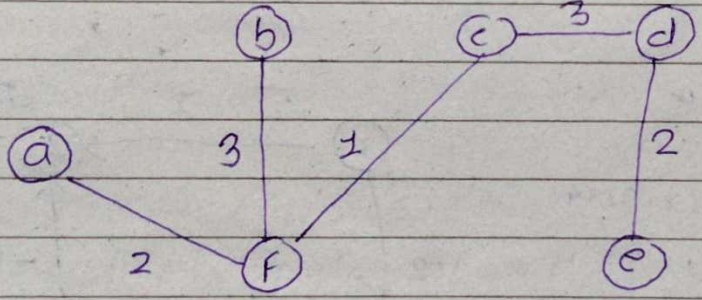


choose

a → 4 (2)

f → 3 (1)

$\Rightarrow$



$2+3+1+3+2$

$= 11$

Algorithm:

```
prims (G, ω, r) {
For each u ∈ G(V)
    u.key = ∞
(parent) π(u) = Nil
    key(r) = 0
    Q = G(V)          ⟶ O(V)
```

```
while Q ≠ ϕ                                  → O(V)
  u = Extractmin(Q)                          → heap → O(logV)
  for each v ∈ G.Adj[u]                      → O(V) O(E)
  {                                            dense  complete
    if v ∈ Q and w(u,v) < v.key                1 unit.
      π(v) = u                                 VlogV
      key[v] = w(u,v)
  }
}
```

	a	b	c	d	e	f
key	0	∞	∞	∞	∞	∞
parent		4 3	1	3	4 2	2
parent	NJ	a f	f	c	f d	a

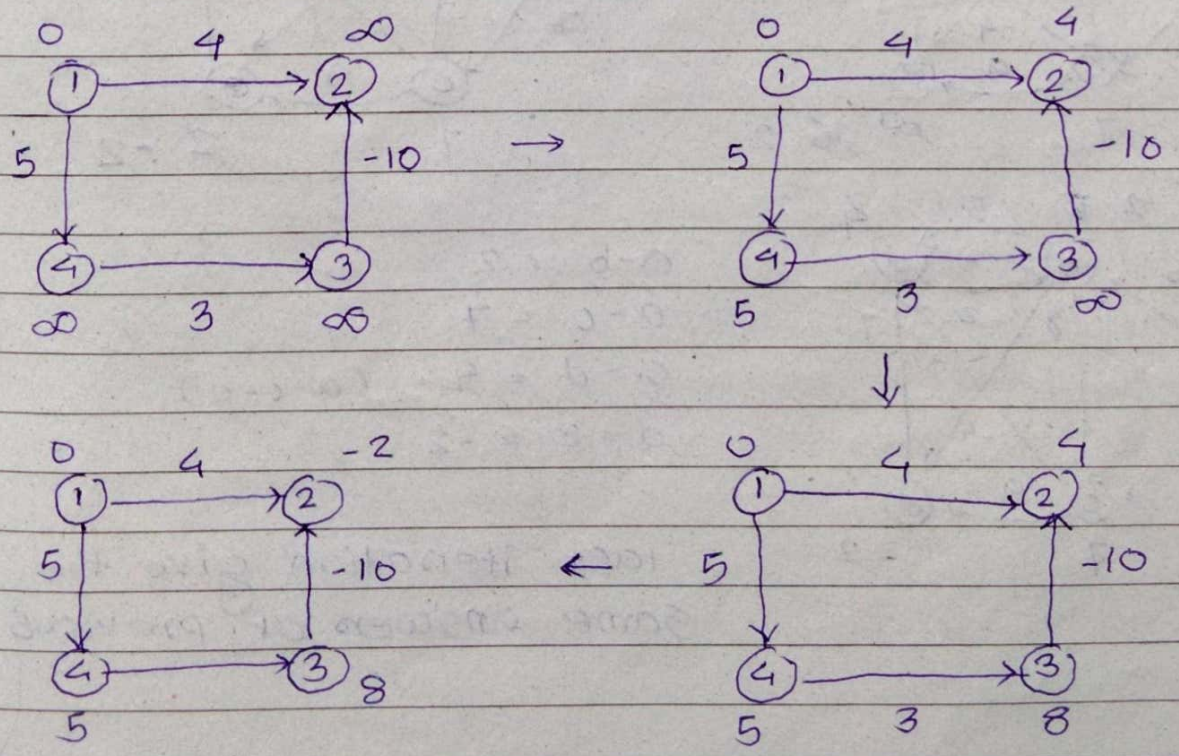
$Q = \{a, b, c, d, e\}$

extract min until $Q = \phi$

complexity =

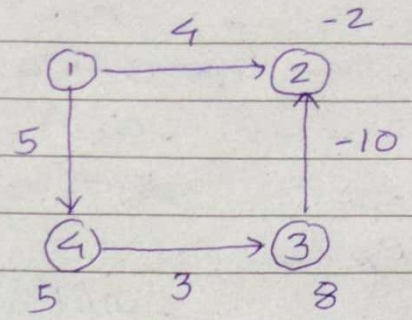
$V^2 \log V \;/\; (V+E)\log V$

* Bellman Ford Algorithm (Relax upto V-1 time)



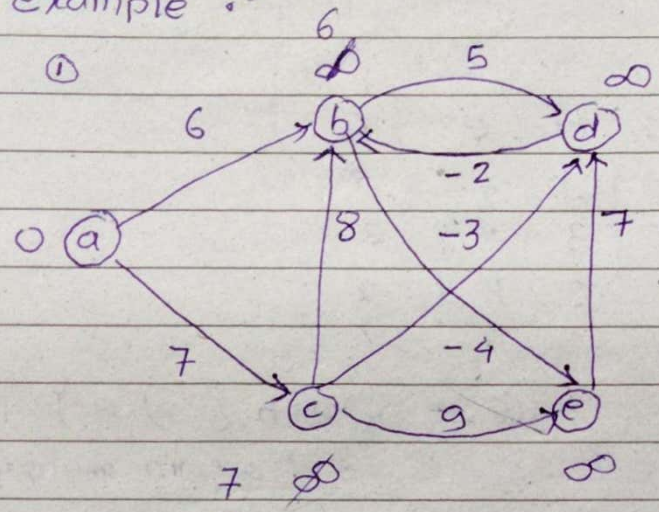
$d(v) > d(u) + w(u,v)$

$d(v) = d(u) + w(u,v)$

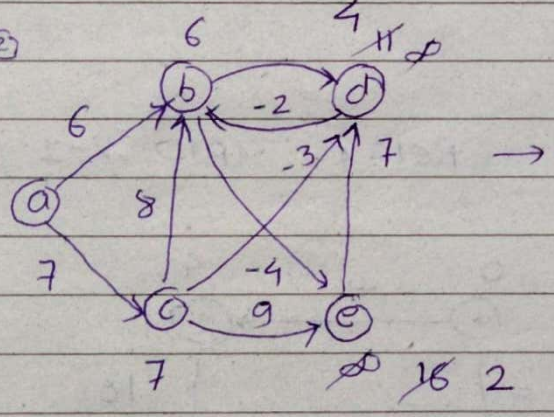


- it always gives the right answer for all negative edge present.
- greater complexity than dijkstra algo.

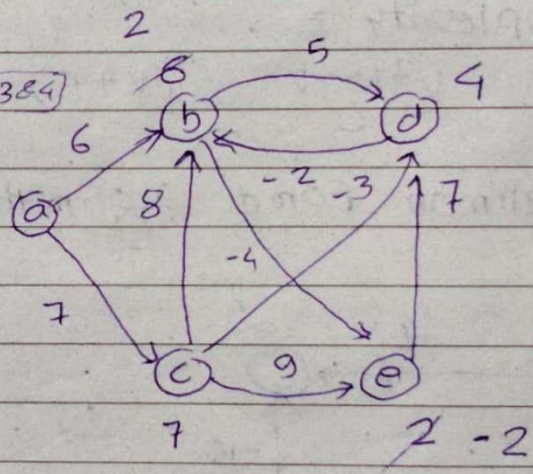
example :-

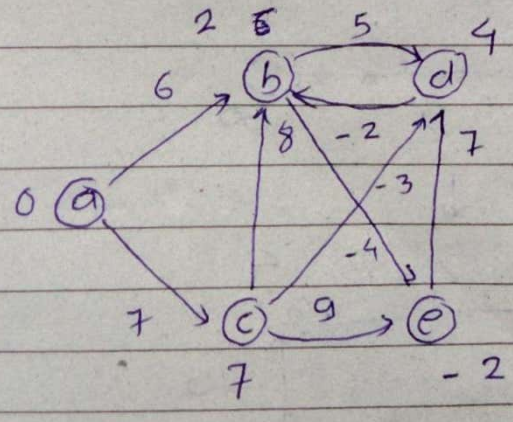


(a,b) (a,c) ~~(b,e)~~ (c,b)
(b,d) (c,e) ~~(d,e)~~ (ed)
(db) (be) (cd)



→

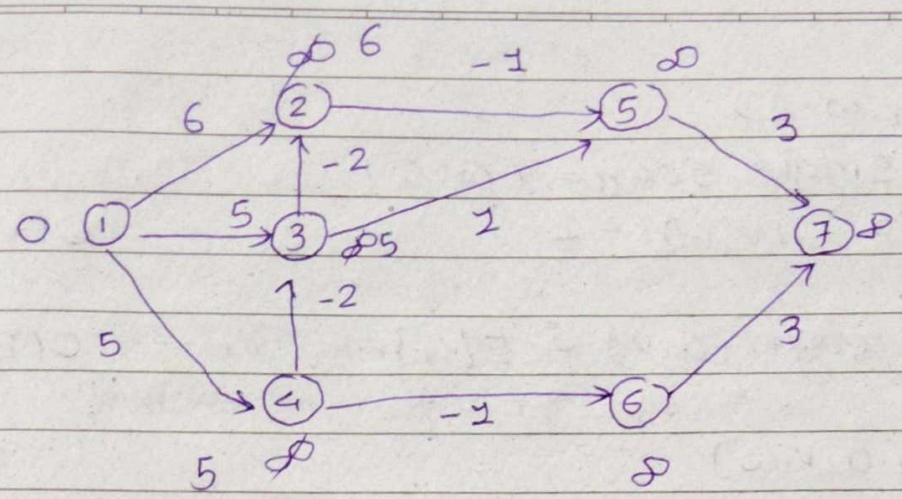




a-b = 2
a-c = 7
a-d = 4 (a-c-d)
a-e = -2

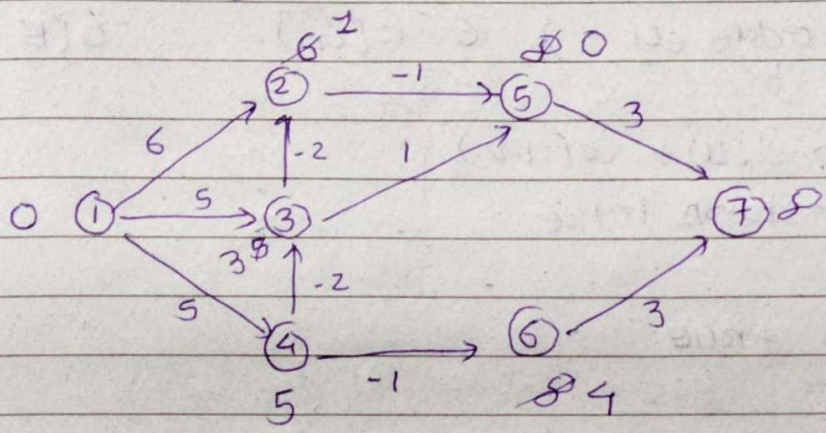
last iteration give the same answer of previous.

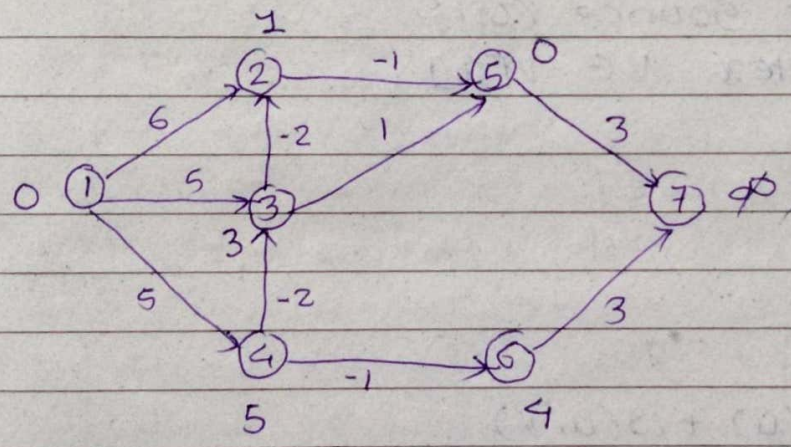
Q.

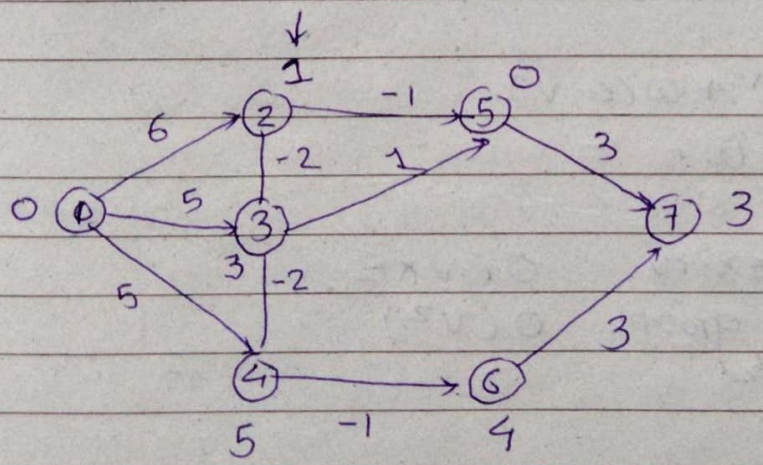


(1,2) (5,7)
(1,3) (6,7)
(1,4)
(4,3)
(3,2)
(2,5)
(3,5)
(4,6)

(2)







1 - 2 → 1
1 - 3 = 3
1 - 4 = 5
1 - 5 = 0 (1-4-3-2-5)
1 - 6 = 4
1 - 7 = 3

diff betn dijkstra & bellman.

→ Bellman (G, ω, s)
initialize single source (G, s)
for i = 1 to |V(G)| - 1 — O(V)
do
for each edge (u, v) ∈ E[G] — O(E)
do
Relax(u, v, ω)
for each edge (u, v) ∈ E[G] — O(E)
do
if d(v) > d(u) + ω(u, v)
then Return False
}
return true.

→ initialize single source (G, s)
for each vertex v ∈ V[G]
do
d[v] ← ∞
π[v] ← Nil
d[G] ← 0

→ Relax (u, v, ω)
if d(v) > d(u) + ω(u, v)
then
d[v] = d(u) + ω(u, v)
π[v] ← u

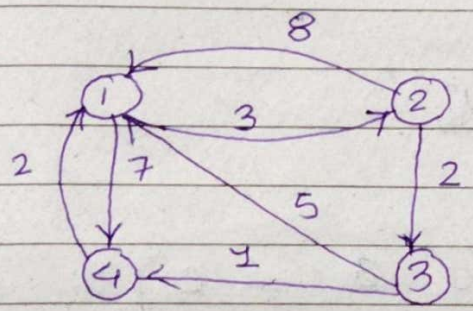
* overall complexity O(V × E
complete graph O(V²).

* Floyd warshall Algorithm

$$d_{ij}^{K} = \min\left(d_{ij}^{K-1}, d_{iK}^{K-1} + d_{Kj}^{K-1}\right)$$

- dynamic programming algo
- use to find the shortest paths beth all pairs of vertices in weighted graph.
- ex:-

1)



$$\Rightarrow D_0 = \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix}$$

$$D_1 = \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & (2) & (15) \\ 5 & (8) & 0 & (1) \\ 2 & (5) & (\infty) & 0 \end{bmatrix}$$

$D_{23} = \min(d_{23}, d_{21} + d_{13})$
$(2, 8+\infty) \rightarrow 2$

$D_{24} = \min(d_{24}, d_{21} + d_{14})$

$D_{32} = \min(d_{32}, d_{31} + d_{12})$

$D_{34} = \min(d_{34}, d_{31} + d_{14})$

$D_{43} = \min(d_{43}, d_{41} + d_{13})$

$D_{42} = \min(d_{42}, d_{41} + d_{12})$

$$D_2 = \begin{bmatrix} 0 & 3 & (5) & (7) \\ 8 & 0 & 2 & 15 \\ (5) & 8 & 0 & (1) \\ (2) & 5 & (7) & 0 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} 0 & (3) & 5 & (6) \\ (7) & 0 & 2 & (3) \\ 5 & 8 & 0 & 1 \\ (2) & (5) & 7 & 0 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & & & 6 \\ & 0 & & 3 \\ & & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \rightarrow D_4 = \begin{bmatrix} 0 & (3) & (5) & 6 \\ (5) & 0 & (2) & 3 \\ (3) & (6) & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix}$$

$d_4 = d_2$

Algorithm :-

Floyd Warshall (w) — TC → O(V3), SC → O(V2)

n = w.rows

$D^0 = w$

for k = 1 to n

let $D^k = D^k_{ij}$ be a new matrix

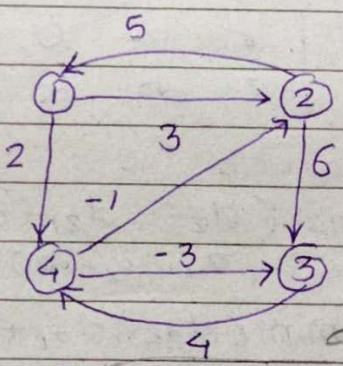
for i = 1 to n

for j = 1 to n

$d_{ij} = \min(d^{k-1}_{ij}, d^{k-1}_{ik} + d^{k-1}_{kj})$

return D^n

2)



$$D_0 = \begin{bmatrix} 0 & 3 & \infty & 2 \\ 5 & 0 & 6 & \infty \\ \infty & \infty & 0 & 4 \\ \infty & -1 & -3 & 0 \end{bmatrix}$$

as it is D_1 (k=1) $d_{ij}, d_{ik} + d_{kj}$

$$D_1 = \begin{bmatrix} 0 & 3 & \infty & 2 \\ 5 & 0 & (6) & (7) \\ \infty & (\infty) & 0 & (4) \\ \infty & (-1) & (-3) & 0 \end{bmatrix}$$

$D_{23} = \min(6, 5+\infty) = 6$

$D_{24} = \min(\infty, 5+2) = 7$

$D_{32} = \min(\infty, \infty+3) = \infty$

$D_{34} = \min(4, \infty+2) = 4$

$D_{42} = \min(-1, \infty+3) = -1$

$D_{43} = \min(-3, \infty+\infty) = -3$

$$D_2 = \begin{bmatrix} 0 & 3 & (9) & (2) \\ 5 & 0 & 6 & 7 \\ (\infty) & \infty & 0 & (4) \\ (4) & -1 & (-3) & 0 \end{bmatrix}$$

$D_{13} = \min(\infty, 3+6) = 9$

$D_{14} = \min(2, 3+7) = 2$

$D_{31} = \min(\infty, \infty+5) = \infty$

$D_{34} = \min(4, \infty+2) = 4$

$D_{41} = (\infty, -1+5) = 4$

$D_{43} = (-3, -1+\infty) = -3$

$$D_3 = \begin{bmatrix} 0 & (3) & 9 & (2) \\ (5) & 0 & 6 & (7) \\ \infty & \infty & 0 & 4 \\ (4) & -1 & -3 & 0 \end{bmatrix}$$

$D_{12} = (3, 9+\infty)$

$D_{14} = (2, 9+4)$

$D_{21} = (5, 6+\infty)$

$D_{24} = (7, 6+4)$

$D_{41} = (4, -3+\infty)$

$D_{42} = (-1, -3+\infty)$

$D_4 =$

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

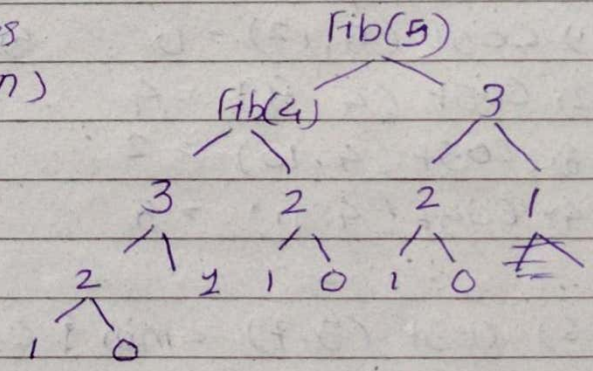
$D_{12} = (3, 2+(-1) = 1$
$D_{13}\ (9, 2+(-3) = -1$
$D_{21} = (5, 7+4)$
$D_{23} = (6, 7+-3) = 4$
$D_{31} = (8, 4+4)$
$D_{32} = (\infty, 4-1)$

- Dynamic programming - always give us optimal soln.

1) proincipal of optimality - complexity is large than the greedy method.
2) overlapping subproblem
3) optimal substructure
4) memorization.

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

$n = 5$ 15 times
Complexity $= \frac{2^n}{2} = O(2^n)$



In this we need to minimize this Memorization

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

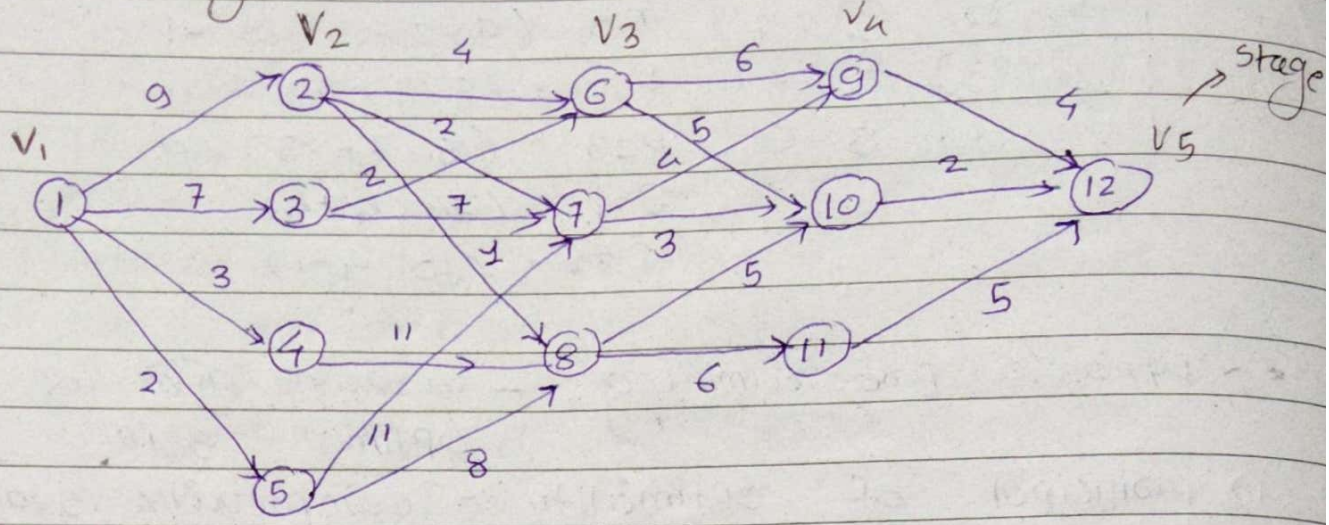
n+1

Fib(0)	(1)	(2)	(3)	(4)
0	1	1	2	3

In memorization we store the value & use this later.

O(n)

- Multistage Graph problem



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	213	7	6	8	8	10	10	10	12	12	12	12

$$cost(i,j) = \min \{ c(j,i) + cost(i+1, l)^2 \}$$
$$<i,l> \in E$$
$$l \in V_{i+1}$$

↑ stage

1) $Cost(5,12) = 0$

2) $Cost(4,9) = 4$

3) $Cost(4,10) = 2$

4) $Cost(4,11) = 5$

5) $Cost(3,6) = \min \{ c(6,9) + cost(4,9), c(6,10) + cost(4,10) \}$
$= \min \{ 6+4, 5+2 \}$
$= \min \{ 10, 7 \}$
$= 7$

6) $Cost(3,7) = \min \{ c(7,9) + cost(4,9), c(7,10) + cost(4,10) \}$
$= \{ 4+4, 3+2 \}$
$cost(3,7) = 5$

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

8) cost(2,2) = min { c(2,6) + cost(3,6)
(2,7) (3,7)
(2,8) (3,8)
= min { 4+7, 2+5, 1+7
cost(2,2) = 7 v=7.

9) cost(2,3) = min { c(3,6) + cost(3,6)
(3,7) + cost(3,7)
= min { 2+7, 7+5)
= 9 v=6

10) cost(2,4) = min { c 11 +7 = 18

11) cost(2,5) = min { c(5,7) + cost(3,7)
(5,8) + cost (3,8
= { 11+5, 8+7)
= 15 v=8.

12) 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 v=2,3

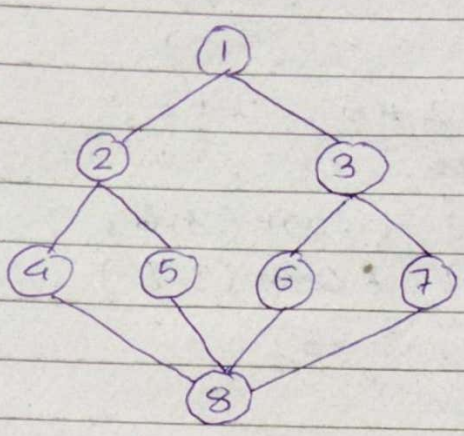
For (2)
(1,1) = 2
(2,2) = 7 → Vertex
(3,7) = 10
(4,10) = 12

1-2-7-10-12

For(2)
(1,1) = 3
(2,3) = 6
(3,6) = 10
(4,10) = 12

1-3-6-10-12

* Depth First Search → use stack.



1	2	3	4	5	6	7	8
1	1	01	01	1	1	1	1

initially it is 0 and when visited it become 1

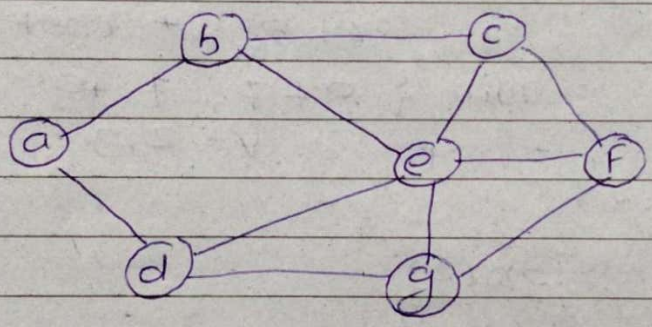
adjacent

u = 1 → (2,3)
u = 2 → (1,4,5)
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)

7
3
6
5
8
4
2
1

O/P → 1,2,4,8,5,6,3,7

Ex-1)



d
g
F
e
c
b
a

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

a, b, c, e, F, g, d

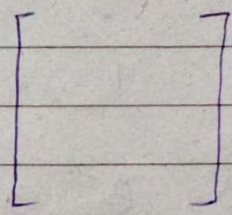
Algorithm :-

```
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 :-

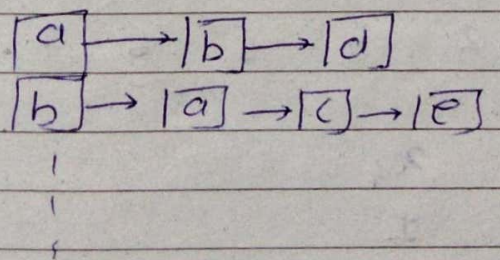
1) Adjacency matrix



n x n → in this we require 2 for loop for traversing

$\therefore O(V^2)$

2) Adjacency list



In this vertex is at least connect to 1 vertex.

→ 2V

And we make one array to make visited 1

→ E

$O(V+E)$

* 0/1 knapsack problem

↳ Solve by greedy as well as dynamic

$m = 50$

W	10	20	30
P	60	100	120
$\frac{P}{W}$	6	5	4

$TC = 2^n$

↓ minimizes

0 0 0	— 0
0 0 1	— 120
0 1 0	— 100
1 0 0	— 60
1 0 1	— 180
1 1 0	— 160
0 1 1	— 220
1 1 1	— X

→ using dynamic programing

$m=8 \quad n=4 \quad P=\{1,2,5,6\} \quad W=\{2,3,4,5\}$

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

x_1	x_2	x_3	x_4
0	1	0	1

Q.2) $\omega = 7 \quad n = 4$

x_1	x_2	x_3	x_4
0	1	1	0

profit = {1, 4, 5, 7}

weight = {1, 3, 4, 5}

$$KS(n,\omega) = \begin{cases} 0 & n = 0, \omega\theta = 0 \\ KS(n-1,\omega) & \omega t(n) > W \\ \max \begin{cases} KS(n-1,\omega) \\ KS(n-1(W-\omega t(n))+P(n)) \end{cases} \end{cases}$$

W	P	$\omega=0$	$\omega=1$	$\omega=2$	$\omega=3$	$\omega=4$	$\omega=5$	6	7
n=0	0	0	0	0	0	0	0	0	0
n=1	1	1	0	1	1	1	1	1	1
n=2	3	4	0	1	1	4	5	5	5
n=3	4	5	0	1	1	4	5	6	6
n=4	5	7	0	1	1	4	5	7	8
~~n=5~~									9

$KS(1,2) = \max \begin{cases} KS(0,2) \\ KS(0,(2-1)+1 \\ KS(0,1)+1 \\ \downarrow 0+1 = 1 \end{cases}$ } same for (1,3) (1,4) (1,5) (1,6) (1,7)

$KS(2,1) = KS(n-1, W) \quad \omega t(n) > W$

$= KS(2-1, 1)$

$KS(1,1) \Rightarrow 1$

$KS(2,2) = KS(2-1, 2) \rightarrow KS(1,2) = 1$

$KS(2,3) = \max \begin{cases} KS(n-1), W) \\ KS(n-1, W-\omega t(n)+P(n) \\ (4-1), 7-5)+1 \rightarrow (3,2) \end{cases}$

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

$= \max \{ {}^{1}_{4} = 4$

KS (3,1) = KS (3-1, 1)
KS (2, 1)

KS (3,2) = KS (3-1, 2)
KS (2,2)

KS (3,3) = KS (3-1, 3) → KS (2,3) → 4

KS (3,4) = { KS (n-1), W)
KS (n-1, (W - w+(n)) + p(n)

= { KS (3-1), 4) → (2,4) = 5
KS (3-1) (4-4) + 5 (2,0)+5 = 5
(5-4)

KS (3,5) = KS (2,1) +5) → 1+5 = 6

KS (3,6) = KS (2,2) +5
KS (2.

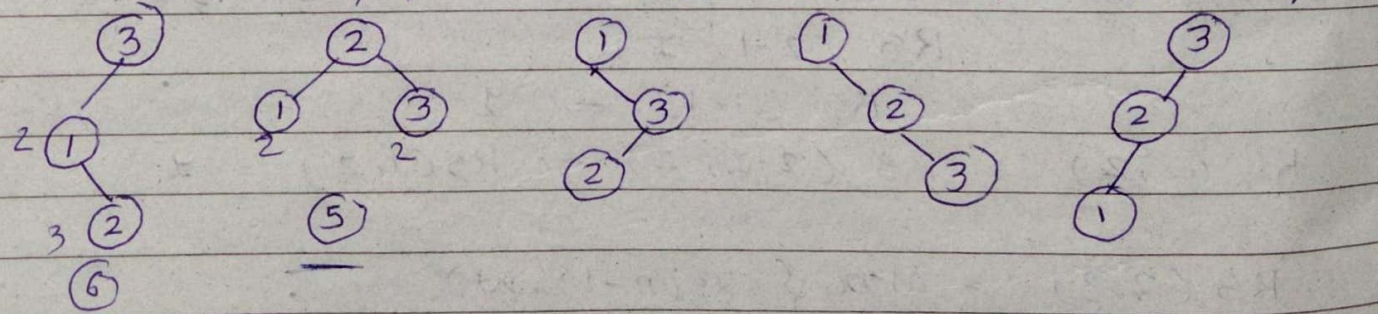
KS (3,7) = { KS (3-1), 5)
KS (3-1) (7-5)+5 (2,2)+5

KS (4,4) = KS (4-1, 4) → KS (3,4) = 5

x_1	x_2	x_3	x_4
0	1	1	0

- Optimal Binary Search Tree (OBST)

1, 2, 3 → key element.
, → This Gives us the minimum cost. comparison

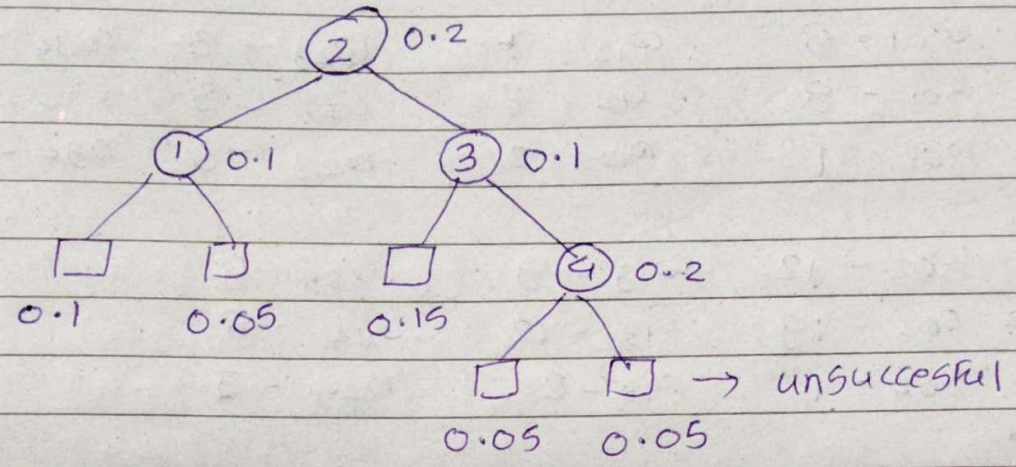


formula $T(n) = \frac{2nCn}{n+1}$ $n=3$

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

= 5

Key	1	2	3	4	
P	0.1	0.2	0.1	0.2	Succesful
q 0.1	0.05	0.15	0.05	0.05	unsuccesful



$= 1\times0.2 + 2\times0.1 + 2\times0.1 + 3\times0.1 + 3\times0.05 + 3\times0.15 + 3\times0.2 + 4\times0.05 + 4\times0.05$

$= 2.5$

~~the~~ optimal Binary search tree are those who ve minimal searching result.

for this we use dynamic programming.

$Keys = \{1, 2, 3, 4\}$

$P_i = \{3, 3, 2, 1\}$

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

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

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

w - weight

c - cost

r - root node

P_1	P_2	P_3	P_4	q_0	q_1	q_2	q_3	q_4
3	3	1	1	2	3	1	1	1

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

$w[0,1] = w[0,0] + P_1 + q_1$
$= 2 + 3 + 3$
$= 8$

$w[1,2] = w[1,1] + P_2 + q_2$
$= 3 + 3 + 1 = 7$

$w[2,3] = w[2,2] + P_3 + q_3$
$= 1 + 1 + 1 = 3$

$w[3,4] = w[3,3] + P_4 + q_4 = 1 + 1 + 1 = 3$

$w[0,2] = w[0,1] + P_2 + q_2 = 8 + 3 + 1 = 12$

$w[1,3] = w[1,2] + P_3 + q_3 = 7 + 1 + 1 = 9$

$w[2,4] = w[2,3] + P_4 + q_4 = 3 + 1 + 1 = 5$

$w[0,3] = w[0,2] + P_3 + q_3 = 12 + 1 + 1 = 14$

$w[1,4] = w[1,3] + P_4 + q_4 = 9 + 1 + 1 = 11$

$w[0,4] = w[0,3] + P_4 + q_4 = 14 + 1 + 1 = 16$

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

$0 < 1 \le 1$

1

$= 0 + 0 + 8$

$= 8$

$$c[0,2] = \min_{0<K\le 2\ (1,2)} \begin{cases} c[0,0] + c[1,2] + \omega[0,2] \\ c[0,1] + c[2,2] + \omega[0,2] \end{cases}$$

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

$$\min \begin{cases} 19 \\ 20 \end{cases} \rightarrow 19 \quad K = 1$$

$$c[1,3] = \min_{1<K\le 3\ (2,3)} \begin{cases} c[1,1] + c[2,3] \\ 0 + 3 \\ c[1,2] + c[3,3] \\ 7 + 0 \end{cases} + \omega[1,3]$$

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

$$c[2,4] = \min_{2<K\le 4\ (3,4)} \begin{cases} c[2,2] + [3,4] \\ 0 + 3 \\ c[2,3] + c[4,4] \\ 3 + 0 \end{cases} + \omega[2,4]$$

$$\min \begin{cases} 3+5 \\ 3+5 \end{cases} = 8 \quad K = 3$$

$$c[0,3] = \min_{0<K\le 3\ (1,2,3)} \begin{cases} c[0,0] + c[1,3] \\ c[0,1] + [2,3] \\ c[0,2] + [3,3] \end{cases} + \omega(0,3]$$

$$\begin{cases} 0 + 12 \\ 8 + 3 = 11 \\ 19 + 0 \end{cases} + 14 = 25 \quad K = 2$$

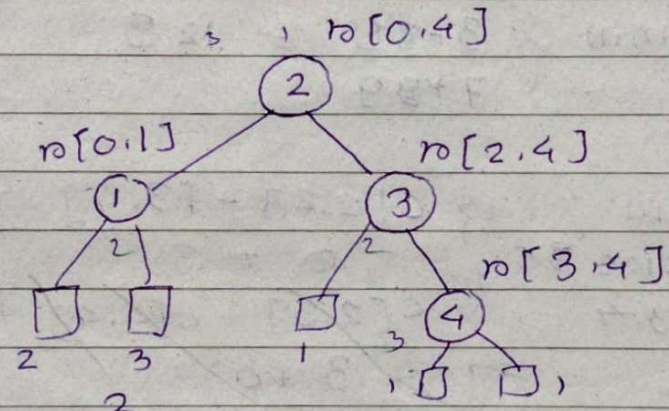
$$c[1,4] = \min_{1<K\le 4,\ 2,3,4} \begin{cases} c[1,1] + c[2,4] \\ c[1,2] + c[3,4] \\ c[1,3] + [4,4] \end{cases} \quad \omega[1,4]$$

$$= \min \begin{cases} 0 + 8 \\ 7 + 3 \\ 12\ 0 \end{cases} + 11 = \cancel{16}\ 19$$

$$c[0,4] = \min_{0<K\le 4,\ 1,2,3,4} \begin{cases} c[0,0] + c[1,4] \\ c[0,1] + c[2,4] \\ c[0,2] + c[3,4] \\ c[0,3] + c[4,4] \end{cases} + \omega[0,4]$$

$$= \min \begin{cases} 0 + 19 \\ 8 + 8 \\ 19 + 3 \\ 25 + 0 \end{cases} + 16 = 16+16 = 32$$

$K = 2$



$$= 1\times3 + 2\times3 + 2\times2 + 4\times3 + 3\times2 + 3\times3 + 3\times1 + 4\times1 + 4\times1$$

$$= 3 + 6 + 4 + 12 + 6 + 9 + 3 + 4 + 4$$

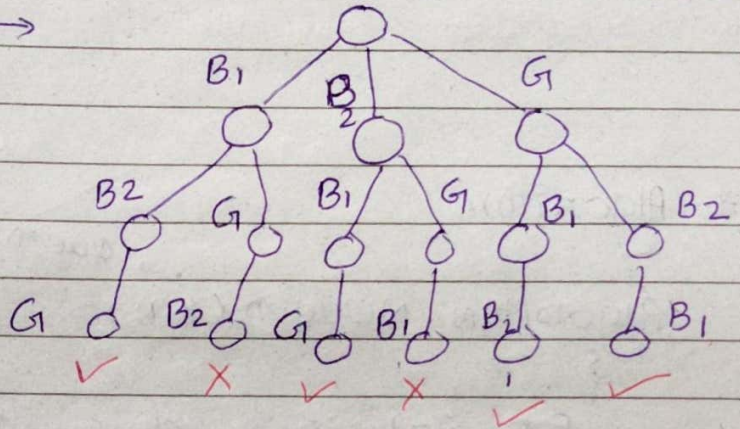
NP-H ; NP-C

- N-Queen problem

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

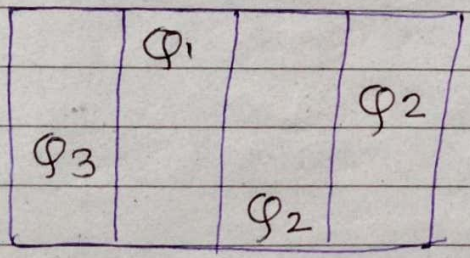
| B_1 | B_2 | G |

→

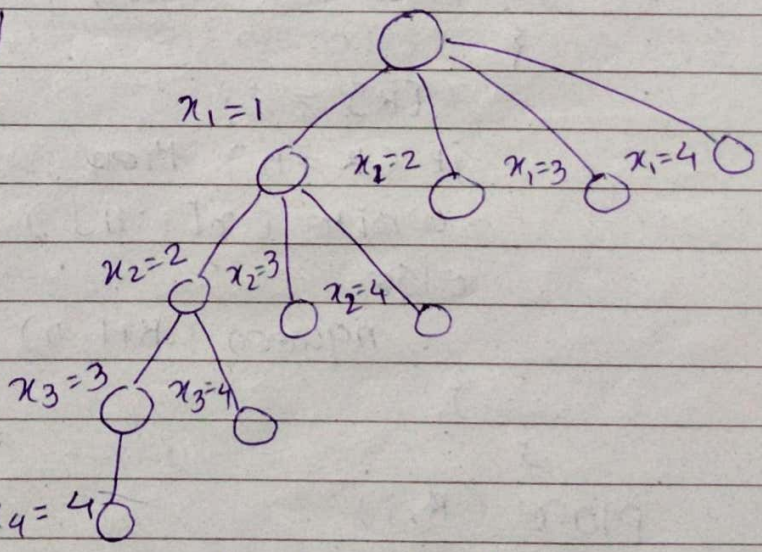


G cannot sit in middle
→ 4 solution

- 4 queen problem



| 2 | 4 | 1 | 3 |



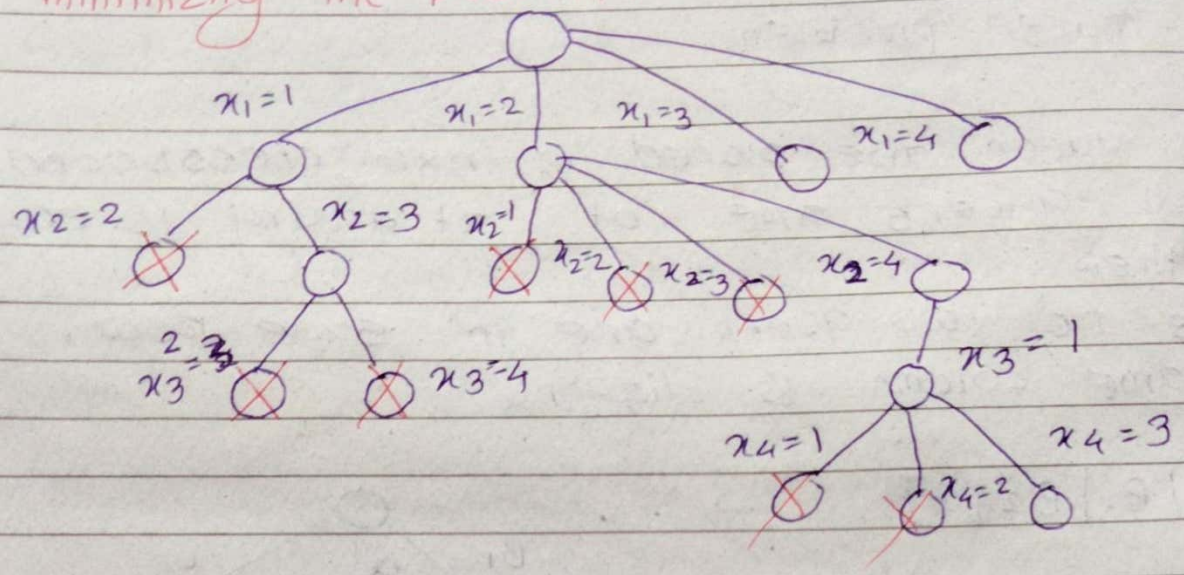
In this there are many possibilities that does not satisfy our problem - we have to minimize it

$= 1\times4 + 4\times3 + 4\times3\times2 + 4\times3\times2\times1$

$= 4 + 12 + 24 + 24$

$= 64$

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
}
```

* 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 some column)
Row 3 : 2 possibility
Row 4 : 1

∴ 4×3×2×1 = 24
N! = 4! = 4×3×2×1 = 24.

Algo :-

```
bool issafe (int row, int column) {
    For (i=0 ; i< 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

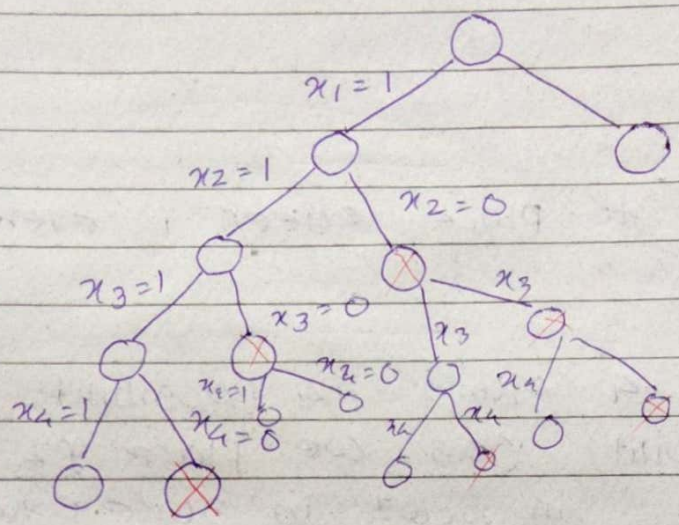
x_1 x_2 x_3 x_4
S = {10, 20, 30, 40}
M = 50
{20, 30}
{10, 40}.

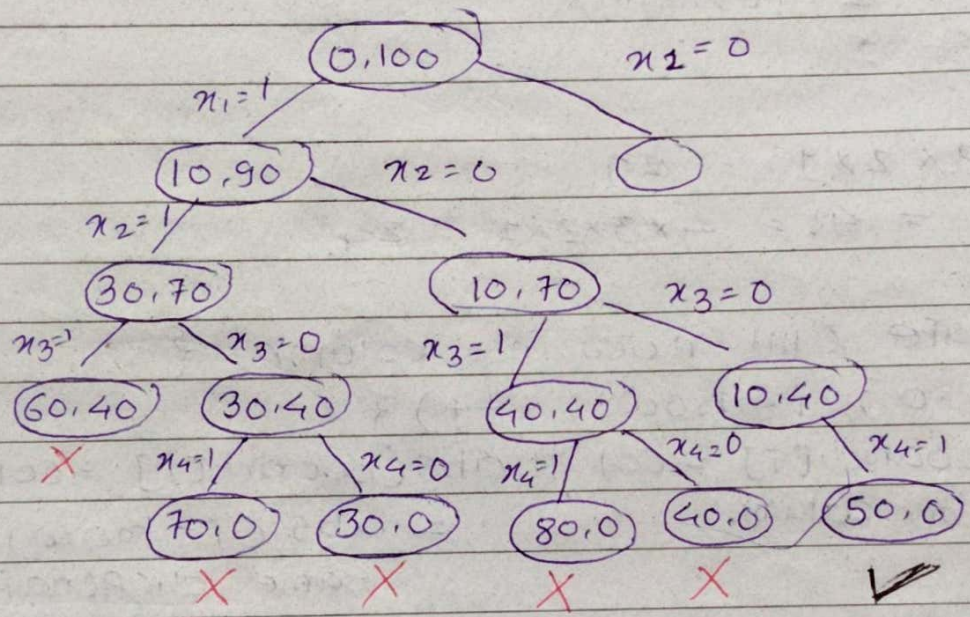
| 0 | 1 | 1 | 0 |

Bounding function.

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

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





Q.2)

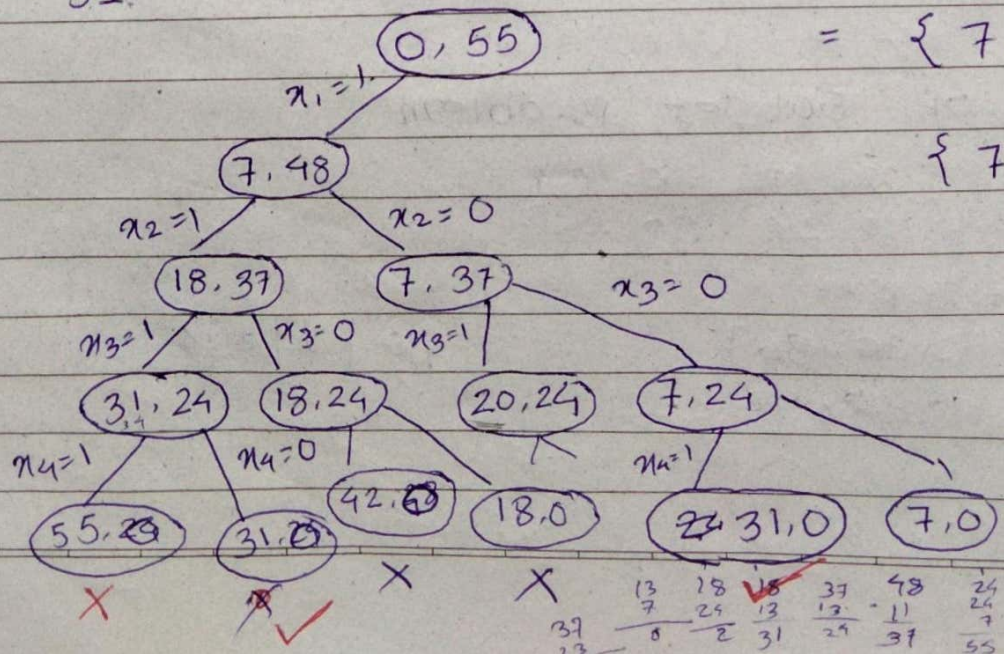
$S = \{7, 11, 13, 24\}$

$m = 31$

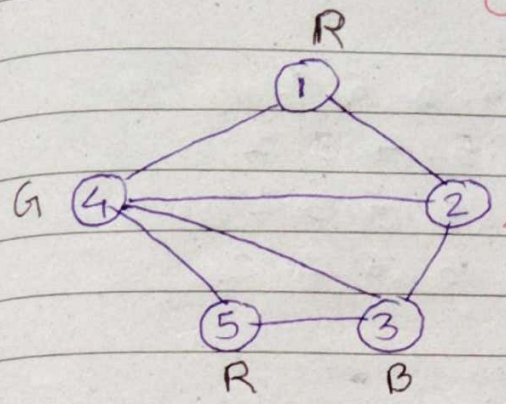
$[1|0|0|1]$

$= \{7, 11, 13\}$

$\{7, 24\}$

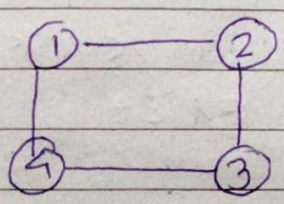


* Graph coloring problem

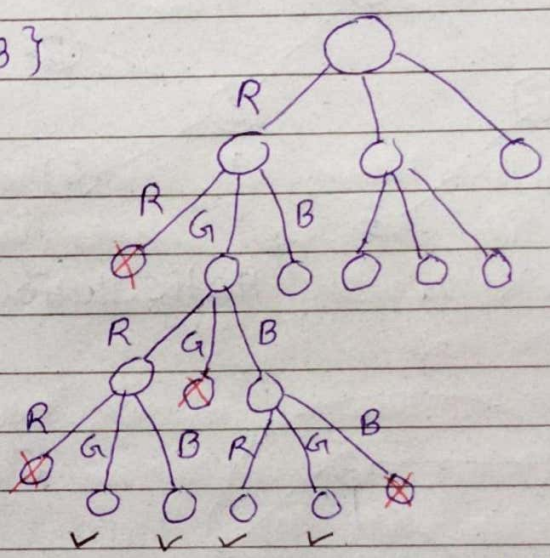


No two adjacent vertex have same color.

ex.



{R, G, B}

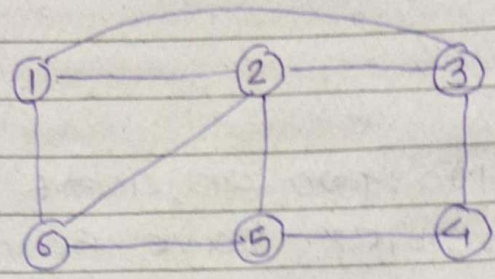


The graph coloring is way to color graph such that vertices, edges & region under some constraint.

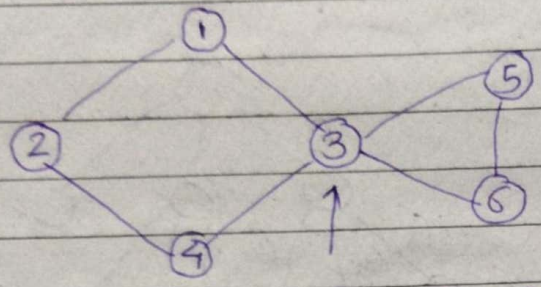
- like no 2 adjacent vertices, edges or region are color with same color.
- if graph is given & some colors are given we want to know whether graph can be colored by using those color or not this problem called as M coloring decision problem.
- if graph is given & we want to know minimum how many color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

complexity → O(N!)

* Hamiltonian cycles

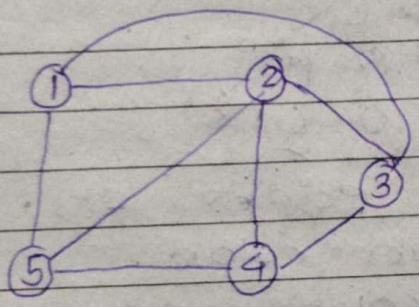


1, 2, 3, 4, 5, 6, 1
1, 2, 6, 5, 4, 3, 1
1, 6, 2, 5, 4, 3, 1
2, 3, 4, 5, 6, 1, 2



→ hamiltonian cycle not exist.

Anticulation point



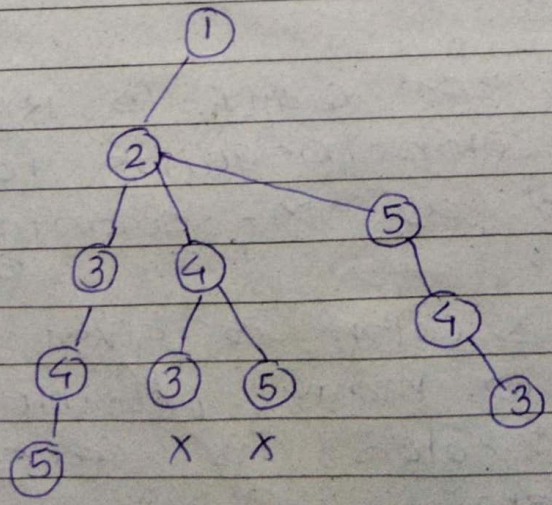
G =

	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

x

1	2	3	4	5
1	2	3	4	5

1, 2, 3, 4, 5, 1
1, 2, 5, 4, 3, 1



```
Algorithm Hamiltonian (K)
{
   do
   {  Nextvertex (K)
      if (x[K] == 0)
          return;
      if (K == n)
          print (x[1:n]);
      else
          Hamiltonian (K+1);
   } while(true);
}

Algorithm Nextvertex (k)
{
   do {
      x[K] = (x[K] + 1) mod (n+1);
      if (x[K] == 0) return;
      if (G[x[K-1]], x[K]] ≠ 0)

      for j = 1 to K-1 do if (x[j] == x[K]) break;
       if (j == K)
       if (K < n or (K == n) && G[x[n], x[1]] ≠ 0
          return;
   } while true;
}
```

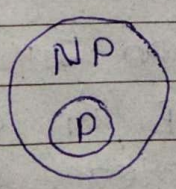
1) Hamiltonian cycle in a graph G is a cycle that visits every vertex of G exactly once & returns to starting vertex.
2) Finding a hamiltonian cycle in graph is well-known NP-complete problem, which means that

there's no known efficient algo to solve it For all types of graphs.
however it can be solved for small or speci types of graph
- HC problem has practical appln in various fields such as logistics, network design, & computer science.

* P, NP, NP-Complete, NP-Hard problems

1) P
- A problem is said to be polynomial problem (P if it can be solved in polynomial time using deterministic algorithm, like $O(n^k)$, where k is constant.
- ex:- linear search, binary S, insertion & merge so
- polynomial problems can be solve & verify in polynomial time.

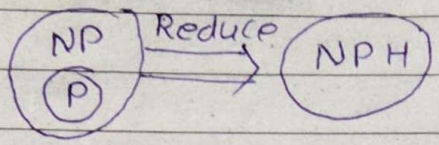
2) NP
- A problem that can not be solved in polynomi time but can be verifiable in polynomial time using Non-deterministic algo is known as NP (non-deterministic polynomial) problem.
- ex. Travelling salesman problem, Sudoku Problem, Scheduling Problem.



P is subset of NP.

NP hard :-

- The problems that can not be solved in polynomial time are called NP hard problem.
- ex - subset sum problem.
- optimization.



NP complete.

- if problem A can be reduced into NP-problem B in polynomial time. Then problem A will be NP-complete problem.
- ex - Graph coloring problem.
  Hamiltonian path problem
- decision making

NP-complete problem can be solved by following approaches;

- Approximation
- Randomization
- parameterization
- Heuristic

NP-hard	NP-Complete.
1) Polynomial time verifn by deterministic machine is not necessary	1) can be verified in polynomial time by a deterministic m/c
2) not a decision problem.	2) exclusively decision problem
3) not all NPH problem are NP-C	3) All NP-C problem are NP-H

4) need not to be a NP- problem.

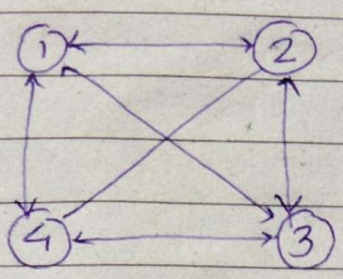
5) ex - Halting problem, Vertex coven Problem.

4) must be NP.

5) ex - Hamiltonial cycle & problem that are NP- Hard.

Traveling Salesman problem.

ex -



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