Unit-1.

1. ASYMPTOTIC NOTATIONS

Asymptotic notation is a notation which is used to make meaningful statements about the efficiency of a program.

Types of Notations:

3 standard notations are,

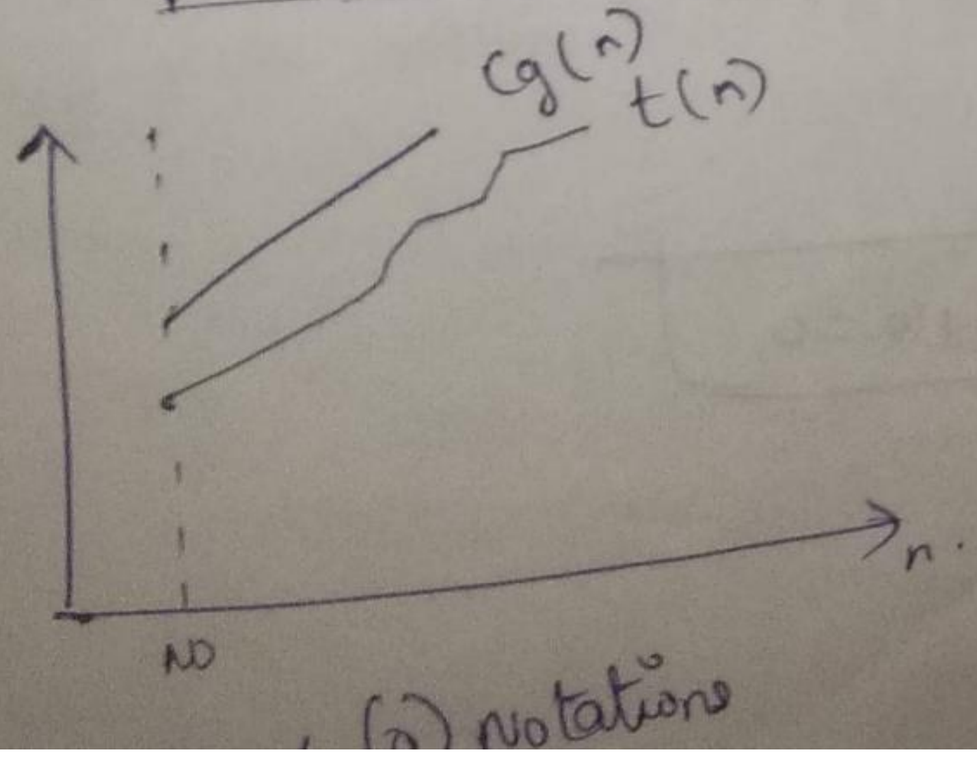
* Big oh (O)
* Big omega (Ω)
* Theta (Θ)

Big oh (O)

Definition:

A function $t(n)$ is said to be in $O(g(n))$ if $t(n)$ is bounded above by sorts constant multiple of $g(n)$ for all large (ie) iff there exists positive constant c and g such that

$$t(n) \leq cg(n) \text{ for all } n > no$$



(a) notations

Ex:

$100n + 5 \in O(n_2)$

So $100n + 5 \leq 100 + n$ (for all $n \geq 5$)

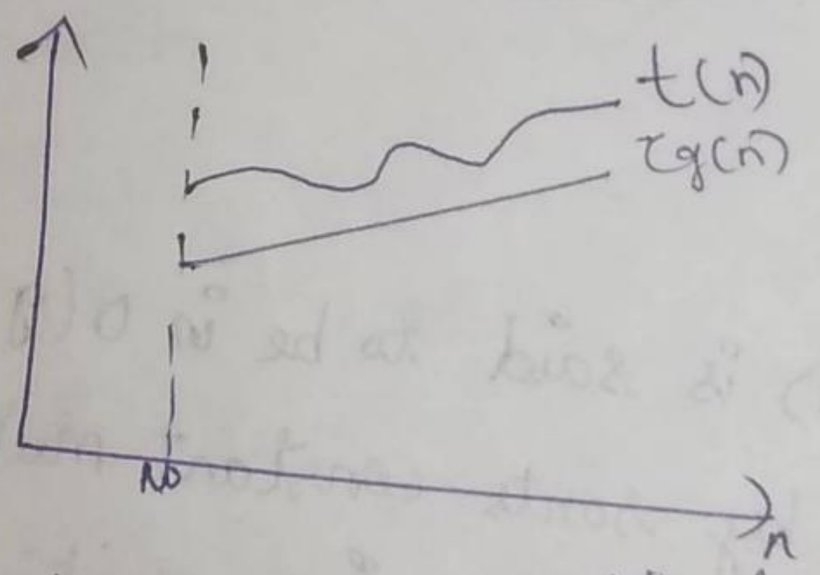
$101n \leq 101n^2$

(and no, use take 101 and 5.

Big omega (Ω) Notation

A function $t(n)$ is said to be $\Omega(g(n))$ denoted $t(n) \in \Omega \cdot (g(n))$ is $t(n)$ is bounded by some constant multiples of $g(n)$ for all large n (ie) if there exists some positive constant (and some non negative integer no such that

$$t(n) \geq cg(n) \text{ for all } n \geq no$$



Big omega notation $f(n) = \Omega(g(n))$

Ex:

$n^1 \in \Omega(n^2)$

$n^3 \geq n^2$ for all $n \geq 0$

we can select

$C = 1$ and $no = 0$

Theta Notations (Θ)

A function $t(n)$ is said to be in $\Theta(g(n))$ denoted $t(n) \in \Theta(g(n))$, if $t(n)$ is bounded both above and below by some positive constant multiples of $g(n)$ for all large n. (ie) if there exist some positive constant c_1 and c_2 and some non-negative integer n_0 such that.

$$c_2 g(n) \leq t(n) \leq c_1 g(n) \text{ for all } n \geq n_0$$

Ex: $\frac{1}{2} n \cdot (n-1) \in \Theta(n^2)$

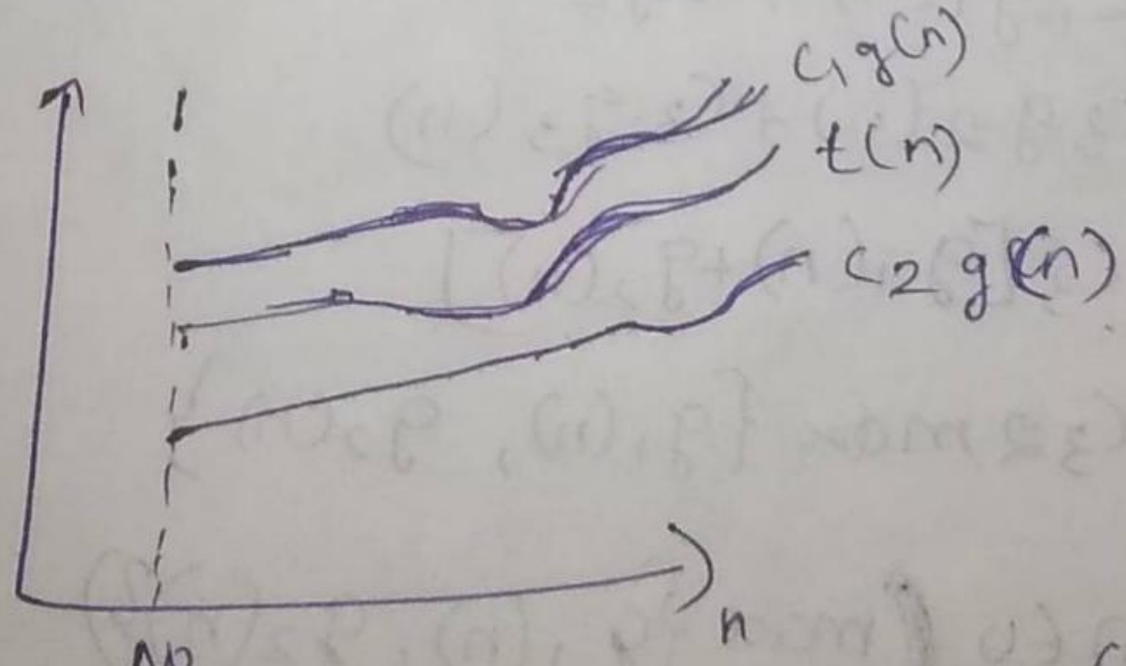
$\frac{1}{2} n \cdot (n-1) = \frac{1}{2} n^2 - \frac{1}{2} n \cdot \leq \frac{1}{2} n^2$ for all $n \geq 0$

(Right in equality is upper bound)

Second we prove that Left inequality (lower bound)

$\frac{1}{2} n(n-1) = \frac{1}{2} n^2 - \frac{1}{2} n \geq \frac{1}{2} n^2 - \frac{1}{2} n \cdot \frac{1}{2} n \cdot = \frac{1}{4} n^2$. (for all $n \geq 2$)

Hence we select $c_1 = \frac{1}{4}$, $c_2 = \frac{1}{2}$.



Big theta notations $t(n) \in \Theta(g(n))$

Useful Property involving Asymptotic Notations :

Theorem :

If $t_1(n) \in O(g_1(n))$ and $t_2(n) \in O(g(n))$ then $t_1(n) + t_2(n) \in O(\max g_2(n)\})$

* Consider 4 arbitary real number a_1, b_1, a_2 and b_2

* If $a_1 \leq b_1$ and $a_2 \leq b_2$ then $(a_1 + a_2) \leq 2 \max \{b_1, b_2\}$

* since $t_1(n) \in O(g(n))$ there exist some constant c_1 and some non negative integer n_1 such that

$$\boxed{t_1(n) \leq c_1 g_1(n) \text{ for all } n \geq n_1}$$

* since $t_2(n) \in O(g(n))$

$$\boxed{t_2(n) \leq c_2 g_2(n) \text{ for all } n \geq n_2}$$

* $c_3 = \max\{c_1, c_2\}$ and consider $n \geq \max\{n_1, n_2\}$

* Adding the 2 inequalities

$$\begin{aligned} t_1(n) + t_2(n) &\leq c_1 g_1(n) + c_2 g_2(n) \\ &\leq c_3 g_2(n) + c_3 g_3(n) \\ &\leq c_3 [g_1(n) + g_2(n)] \\ &\leq c_3 2 \max\{g_1(n), g_2(n)\} \end{aligned}$$

* $\therefore t_1(n) + t_2(n) \in O(\max\{g_1(n), g_2(n)\})$ with the constants c and n_0.

* The asymptotic notations are defined only in terms of the size of the input.

Example - Find the summations of $1^2+2^2+3^2+\cdots n^2$.

Using the Property o $\{f(n) + o(g(n) = o(\max\{f(n), g(n)\})$

$1^2+2^2+3^2+\cdots n^2 = o(\max\{1^2, 2^2, \cdots n^2\})$

$= o(n^2)$

Unit-2

1. Binary Search:

5	7	9	13	32	33	42	54	56	88
0	1	2	3	4	5	6	7	8	9

K = m = 33

$$Mid = \frac{low + high - low}{2}$$

low = 0

high = 9

$$Mid = \frac{0 + 9 - 0}{2} = \frac{9}{2} = 4.5$$

Mid = 4

33 > a[m]

33 > a[4]

low = mid + 1

= 4 + 1

= 5

33	42	54	56	88
5	6	7	8	9

$$Mid = low + \left(\frac{high - low}{2}\right)$$

$$= 5 + \left(\frac{9-5}{2}\right) = 5 + \left(\frac{4}{2}\right) = 7.$$

Mid = 7

$33 < a\ [mid]$

$33 < a\ [7]$

33	42	54
5	6	7

$high = mid - 1$

$= 7 - 1$

$= 6$

33	42
5	6

$$mid = low + \left(\frac{high - low}{2}\right)$$

$$Mid = 5 + \left(\frac{6-5}{2}\right) = 5 + \frac{1}{2} = 5 + 0.5 = 5.5$$

mid = 5

$33 = a\ [mid]$

$33 = a\ [5]$

$33 = 33$

Algorithm

Binary search type a[a], int n, type k

// input : given an array a[a...n-1] in ascending
Order search key k.

// Output : An index of array element that
is equal to k or -1 is no such element.

{

Int

low = 0

high = n

while (low <= high) do

{

Ind mid = $\left(\frac{\text{low} + \text{high} - \text{low}}{2}\right)$

if

(k > a [mid] low = mid + 1);

else

if

(k > a [mid] high = mid - 1);

else

return (mid);

}

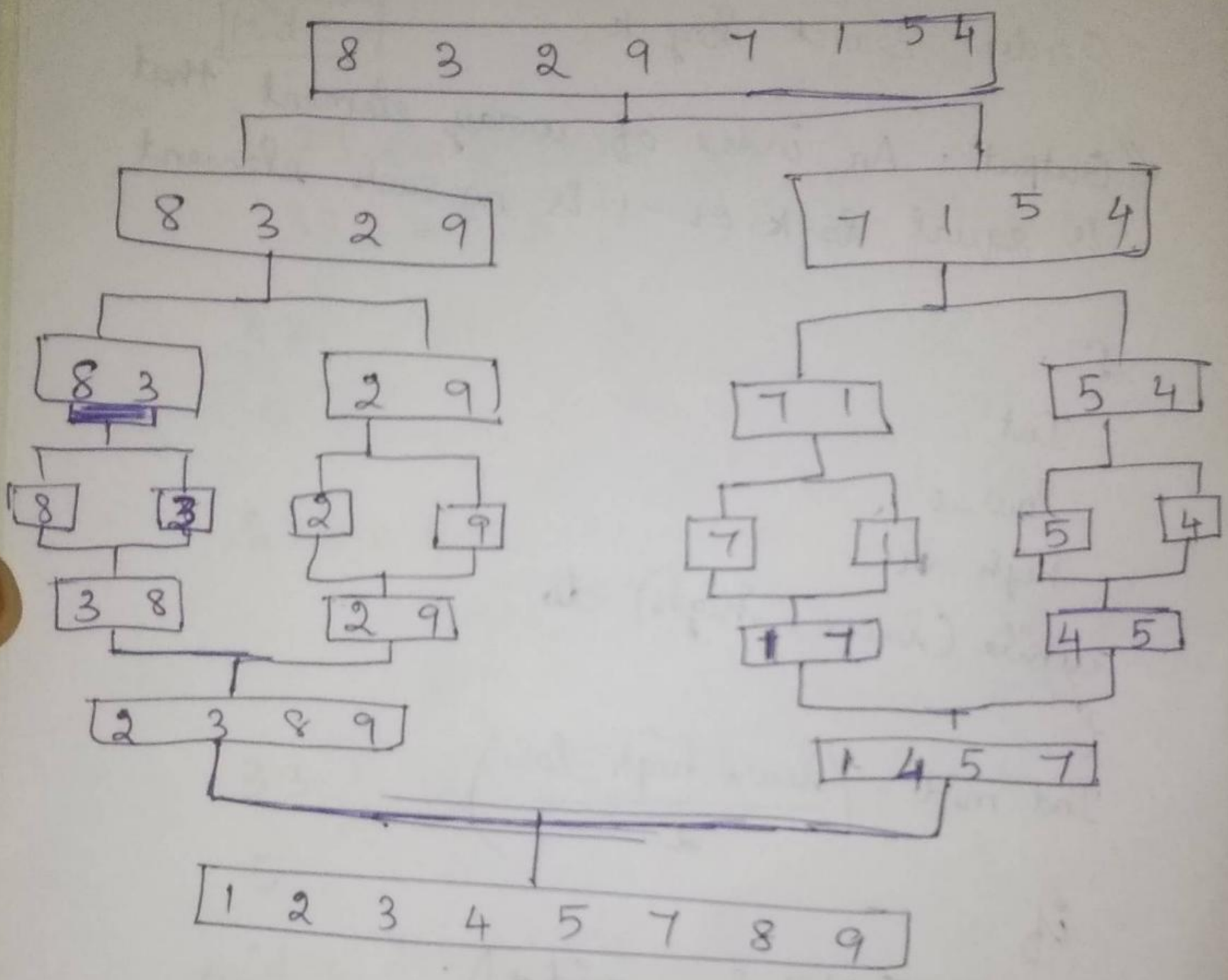
return(0);

}

2. Merge sort: → using an array element

8 , 3 , 2 , 9 , 7 , 1 , 5 , 4

Chang the assending order.



Algorithm:

Algorithm merge (B[0.... p-1], e[0... q-1], A [0... pq-1]);

// merge 2 sorted array into are array.

// input arrays b[0... p-1] and c [0... q-1] bo x sorted

//output sorted array [0 ... p+q-1] of the elements of B and c b[k]=a[i];

```
{
else
b [k] a [j];
}
if ((i) mid)
{
while (j <= high)
{
b[k] = a [j];
```

```
{
else
{
while (i <= mid)
{
b[k] = a [i];
i++;
k++;
}
}
```

```
for (k = low ; k< high; k++)
    a [k] = b[k];
}
}
```

Best case

(A) (nlog2n)

Average case

(B) (nlogn)

3. OOPS SORTING TECHNIQUES

Quick Sort :-

PROCESS:

Consider an array of 6 elements

34 , 99 , 5 , 2 , 57 , 40

ANS:

P - Big
i - Small
j - Middle (mid value)

Pivot all element < Point

0	1	2	3	4	5
34	99	5	2	57	40

1) Pivot element = 34

34 99 5 2 57 40

↓ ↓ ↓

P I J

2) Scan right to left (←) list becomes.

2 99 5 34 57 40

↑ ↓ P ↓

i J

else < 34

3) Scan left to right (→) list becomes.

2 34 5 99 57 40

↓ ↓ ↓

P i j

else > 34

3) Scan Right to left [←] list becomes:

2 5 , 34 99 57 40

Pivot element = 40. else < 34

4) Scan left to Right [→] list becomes.

2 5 , 34 40 57 99

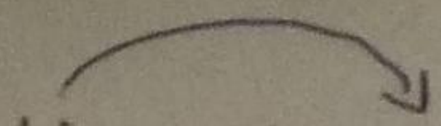
else > 34.

The accending order

2 , 5 , 34 , 40 , 57 , 99.

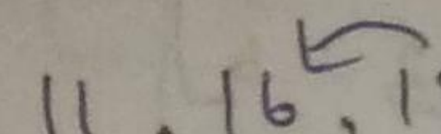
Algorithm:

4. Selection Sort → using array element



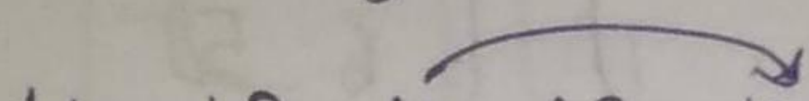
13, 16, 11, 18, 14, 15

selection sort.



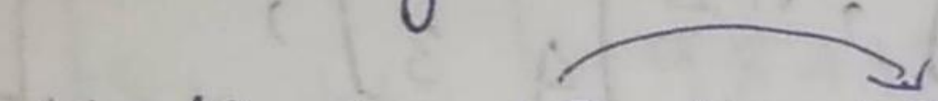
11, 16, 13, 18, 14, 15

Passing 2th



11, 13, 16, 18, 14, 15

Passing 3th



11, 13, 14, 18, 16, 15

Passing 4th

11, 13, 14, 15, 16, 18

Passing 5 : 11, 13, 14, 15, 16, 18

n-1

Sorted array 11, 13, 14, 15, 16, 18

Algorithm:

$$P_1 = (A_{11} + A_{22})(B_{11} + B_{12})$$

$$= \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 5 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 7 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} 12 & 14 \\ 4 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 7 \\ 7 & 3 \end{bmatrix} \begin{bmatrix} 12 & 14 \\ 4 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 12+28 & 14+28 \\ 84+12 & 98+12 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 42 \\ 96 & 110 \end{bmatrix}$$

$$P_2 = (A_{21} + A_{22}) \cdot B_{11}$$

$$= \begin{bmatrix} 2 & 7 \\ 4 & 3 \end{bmatrix} + \begin{bmatrix} 0 & 5 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 12 \\ 6 & 4 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 10+12 & 30+4 \\ 36+0 & 36+0 \end{bmatrix} = \begin{bmatrix} 22 & 34 \\ 36 & 36 \end{bmatrix}$$

$$P_3 = (B_{12} - B_{22}) \cdot A_{11}$$

$$= \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 7 & 0 \\ 6 & 5 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 8 \\ -3 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 0+40 & 0+16 \\ -3-5 & -6-2 \end{bmatrix}$$

$$= \begin{bmatrix} 40 & 16 \\ -8 & -8 \end{bmatrix}$$

$$P_4 = (B_{12} - B_{11}) \cdot A_{22}$$

$$= \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix} - \begin{bmatrix} 5 & 6 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 5 \\ 2 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0 & 5 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 0+4 & 10+2 \\ 0+8 & 10+4 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 12 \\ 8 & 14 \end{bmatrix}$$

$$P_5 = (A_{11} + A_{12}) \cdot B_{22}$$

$$= \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix} + \begin{bmatrix} 3 & 4 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 6 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 6 \\ 12 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 6 & 5 \end{bmatrix} = \begin{bmatrix} 28+36 & 0+30 \\ 84+18 & 0+15 \end{bmatrix}$$

$$= \begin{bmatrix} 64 & 30 \\ 102 & 15 \end{bmatrix}$$

$$P_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$$

$$= \begin{bmatrix} 2 & 7 \\ 4 & 3 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & 9 \\ 9 & 5 \end{bmatrix} \cdot \begin{bmatrix} 12 & 14 \\ 4 & 4 \end{bmatrix} = \begin{bmatrix} 36+36 & 42+36 \\ 111+20 & 126+20 \end{bmatrix}$$

$$= \begin{bmatrix} 72 & 78 \\ 131 & 146 \end{bmatrix}$$

$$P_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})$$

$$= \begin{bmatrix} 3 & 4 \\ 7 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 5 \\ 2 & 1 \end{bmatrix} \cdot \begin{bmatrix} 6 & 2 \\ 8 & 1 \end{bmatrix} + \begin{bmatrix} 7 & 0 \\ 6 & 5 \end{bmatrix} = \begin{bmatrix} 3 & -1 \\ 9 & 0 \end{bmatrix} \cdot \begin{bmatrix} 13 & 2 \\ 12 & 6 \end{bmatrix}$$

$$= \begin{bmatrix} 39-12 & 6-6 \\ 117+0 & 18+0 \end{bmatrix} = \begin{bmatrix} 27 & 0 \\ 117 & 18 \end{bmatrix}$$

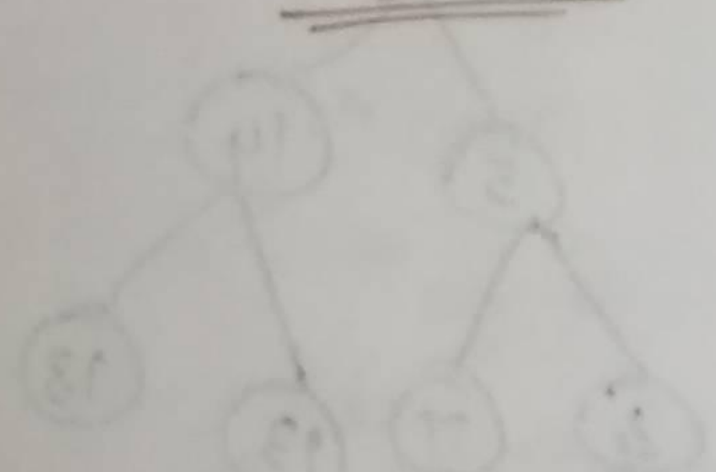
$$P_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})$$

$$= \begin{bmatrix} 2 & 7 \\ 4 & 3 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 5 & 2 \end{bmatrix} \cdot \begin{bmatrix} 5 & 6 \\ 10 & 3 \end{bmatrix} + \begin{bmatrix} 7 & 8 \\ 3 & 4 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 5 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 12 & 14 \\ 4 & 4 \end{bmatrix} = \begin{bmatrix} 12+20 & 14+20 \\ -12+4 & -14+4 \end{bmatrix}$$

$$= \begin{bmatrix} 32 & 34 \\ -8 & -10 \end{bmatrix}$$

Algorithm:



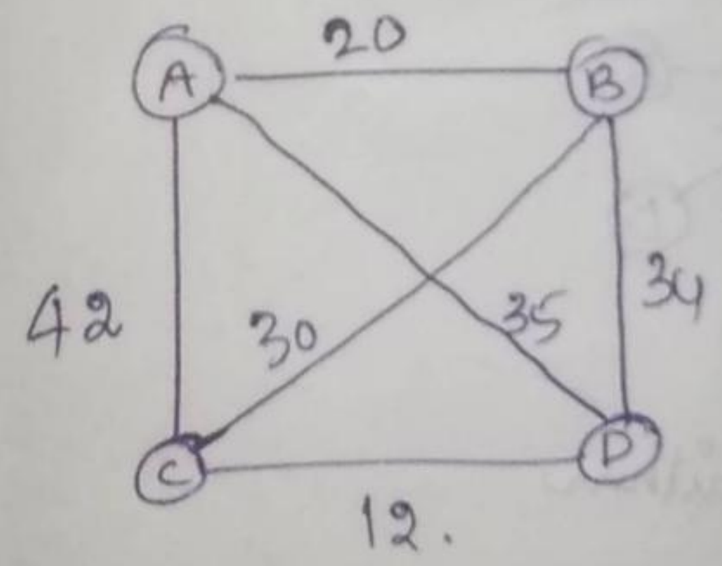
What is Greedy algorithm ?

* In the hard words : A greedy algorithm is an algorithm that follows the Problem solving heuristics of making the locally optimal choice at each stage with the hope of finding a global algorithms.

* Simplify : Choose the best choice that 'reachable' at current state .

Ex:

For better explanation we use old simple problem: Travelling Salesman Problem :

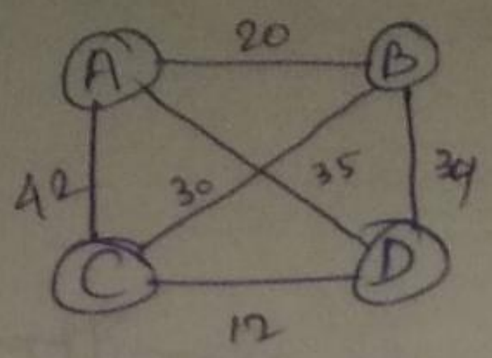


* The Problem is how to travel from city A and visit all city on the map, then back to city A again.

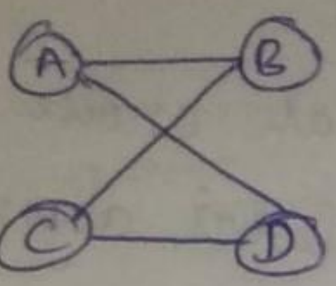
* The rules: you only visit each city once and you can't Pass through any traversal Path.

Sol

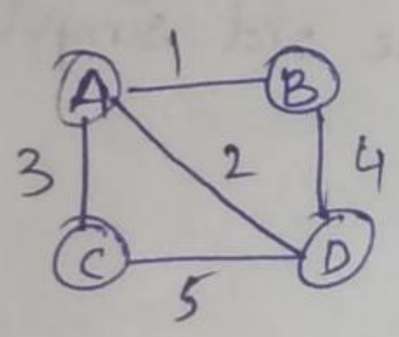
* Find the Shortest Path from city A(start) to any other city.

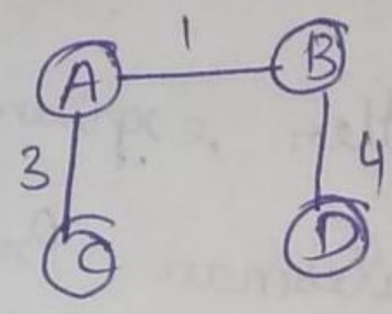


* Because the nearest city is B, so we go to B.

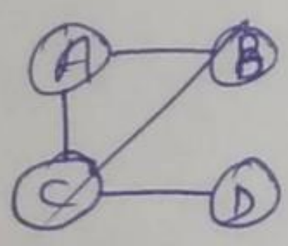


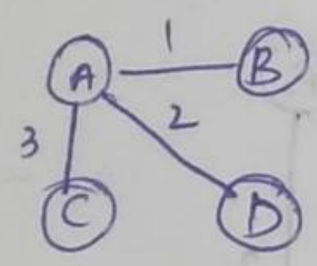
2. Minimum Spaning Tree. (MST)





$= 3+1+4 = 8$

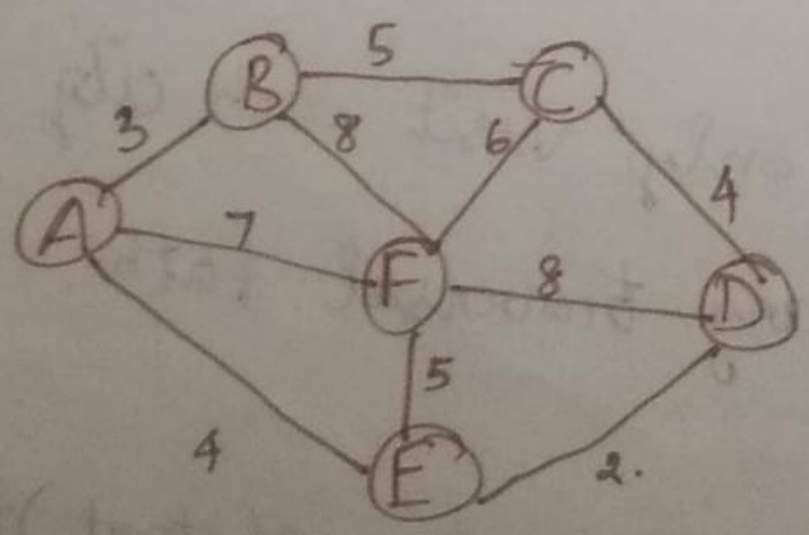




$= 3+2+1 = 6.$

1. Kruskal's Algorithm
2. Prim's Algorithm.

i) MST - Kruskal's Algorithm. Don't form cycle



Step:1

No. of edges : 10

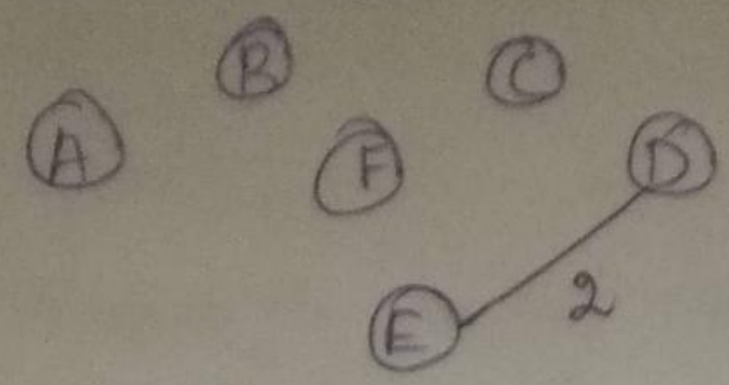
step:2

Vertex pair	Weight
(A,B)	3
(A,F)	7
(A,E)	4
(B,F)	8
(B,C)	5
(C,F)	6
(C,D)	4
(D,F)	8
(D,E)	2
(F,E)	5

Step:3 Vertex pair in ascending order.

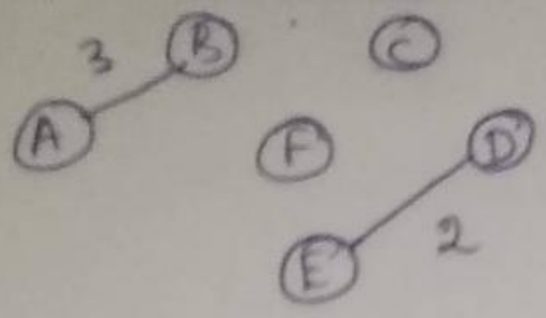
Vertex pair	Weight	Action
(D,E)	2	Accepted
(A,B)	3	Accepted
(A,E)	4	Accepted
(C,D)	4	Accepted
(B,C)	5	Rejected
(F,E)	5	Accepted
(C,F)	6	Rejected
(A,F)	7	Rejected
(B,F)	8	Rejected
(D,F)	8	

(D, E) → 2.



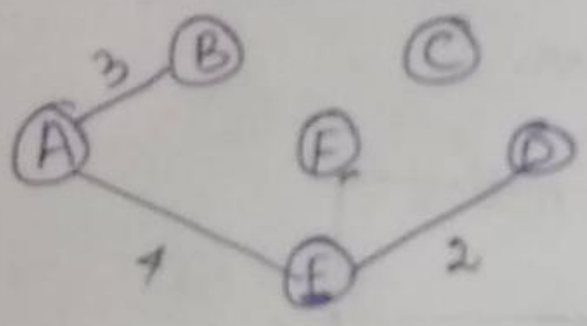
Vertex	Unknown
A	0
B	0
C	0
D	1
E	1
F	0

(A, B) → 3



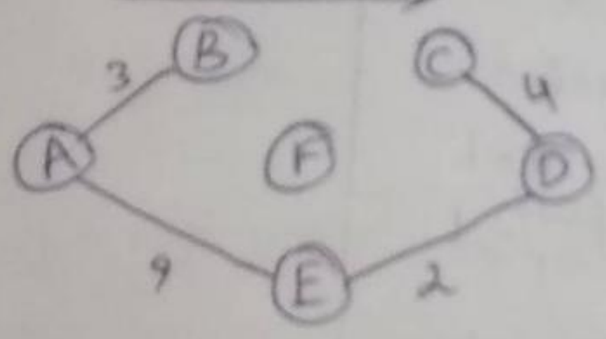
Vertex	unknown
A	1
B	1
C	0
D	1
E	1
F	0

(A, E) → 4



Vertex	unknown
A	1
B	1
C	0
D	1
E	1
F	0

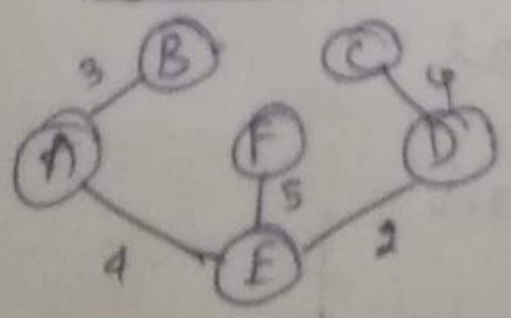
(C, D) → 4



Vertex	unknown
A	1
B	1
C	1
D	1
E	1
F	0

(B, C) is from a cycle, so its is rejected.

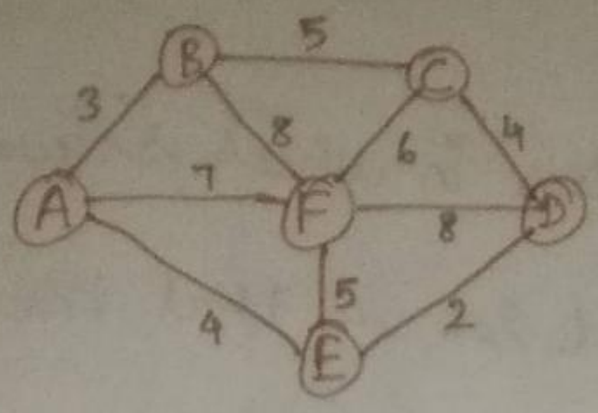
(F, E) → 5



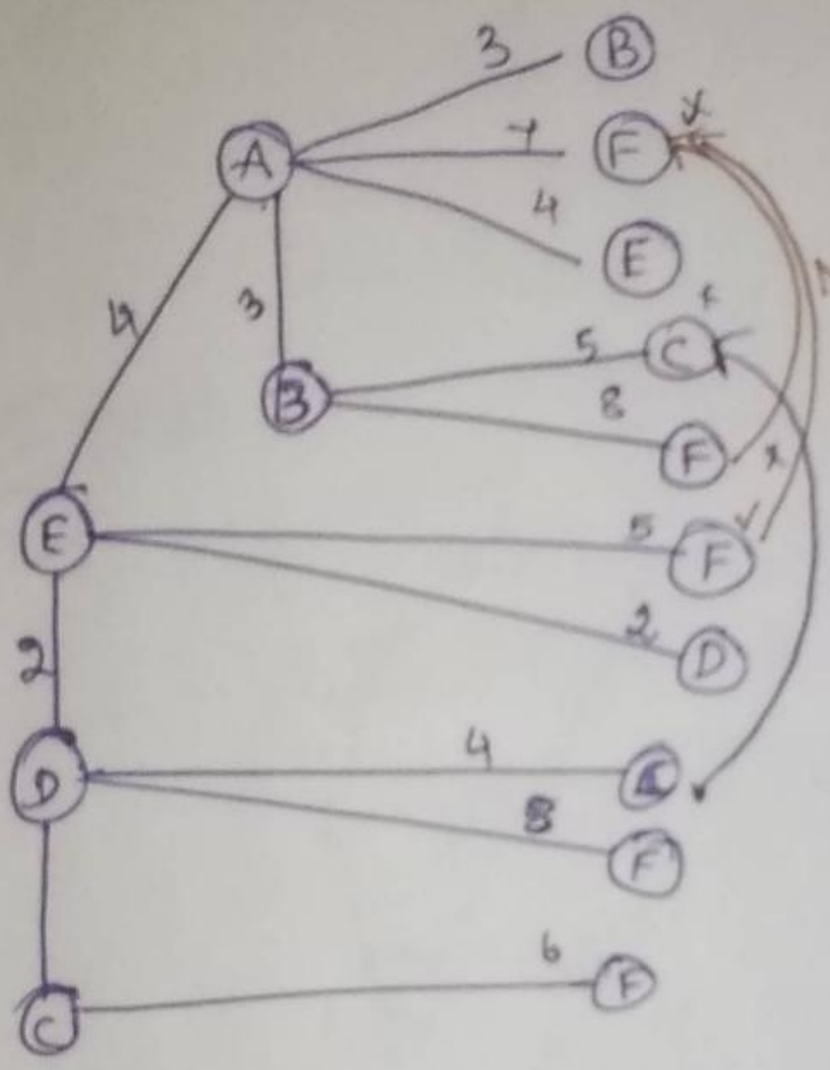
Vertex	unknown
A	1
B	1
C	1
D	1
E	1
F	1

In another pair is from cycle. so, its is rejected

11) MST - Prim's Algorithm



Tree nodes Fringe Nodes



A

Vertex	In tree	dv	Pv
A	1	0	-
B	0	3	A
C	0	∝	-
D	0	∝	-
E	0	4	A
F	0	7	A

B

V	In	dv	Pv
A	1	0	-
B	1	3	A
C	0	5	B
D	0	∝	-
E	0	4	A
F	0	7	B

E

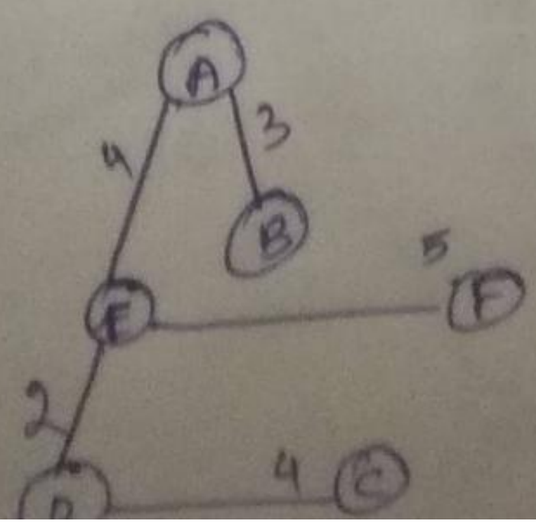
V	In	dv	Pv
A	1	0	-
B	1	3	A
C	0	5	B
D	0	2	E
E	1	4	A
F	0	5	E

D

V	In	dv	Pv
A	1	0	-
B	1	3	A
C	1	4	D
D	1	2	E
E	1	4	A
F	0	5	E

F

V	In	dv	Pv
A	1	0	-
B	1	3	A
C	1	4	D
D	1	2	E
E	1	4	A
F	1	5	E



1. Optimal Merge Pattern - using greedy method.

Merge:

Eg:

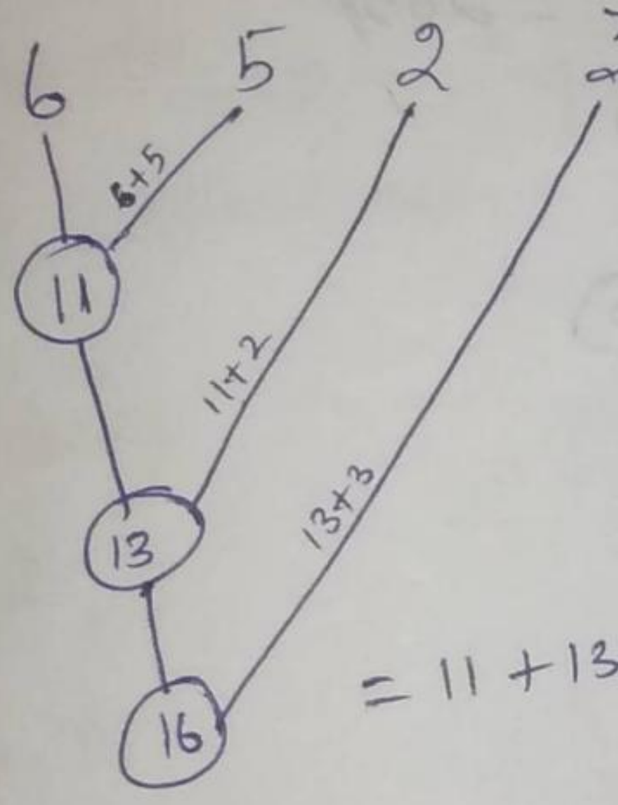
A	B
3	4
5	6
8	7
9	10
11	

C
3
4
5
6
7
8
9
10
11

5+4=9

Maximum

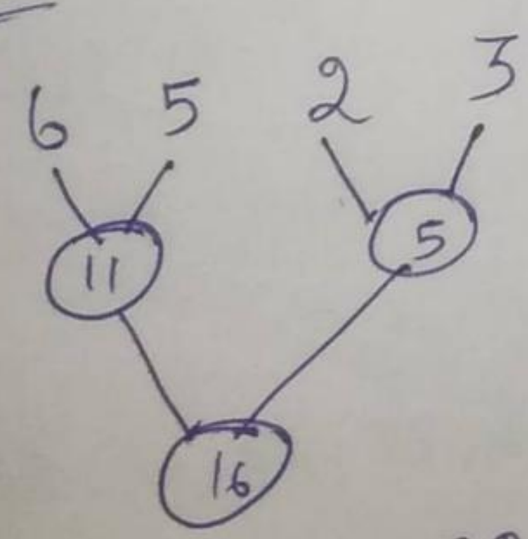
I model



= 11 + 13 + 16 = 40

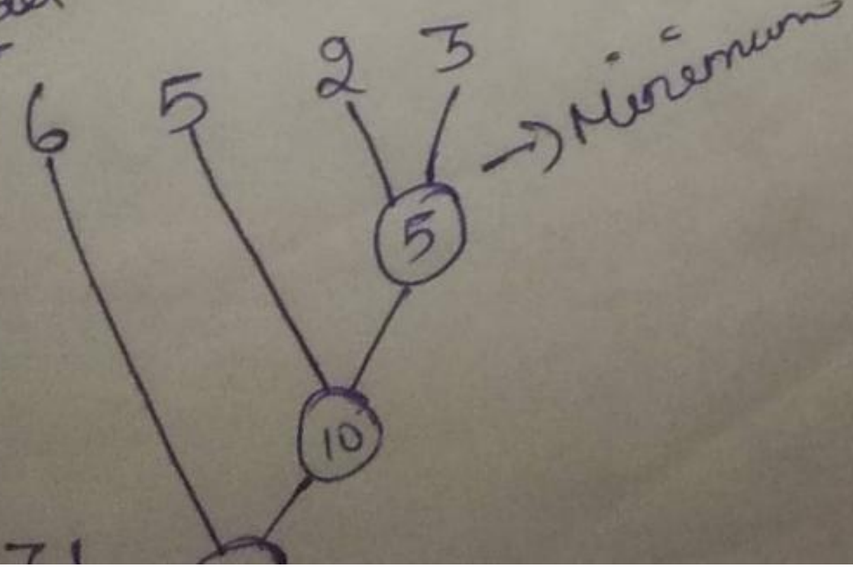
IInd Model

Minimum



11 + 16 + 5 = 32.

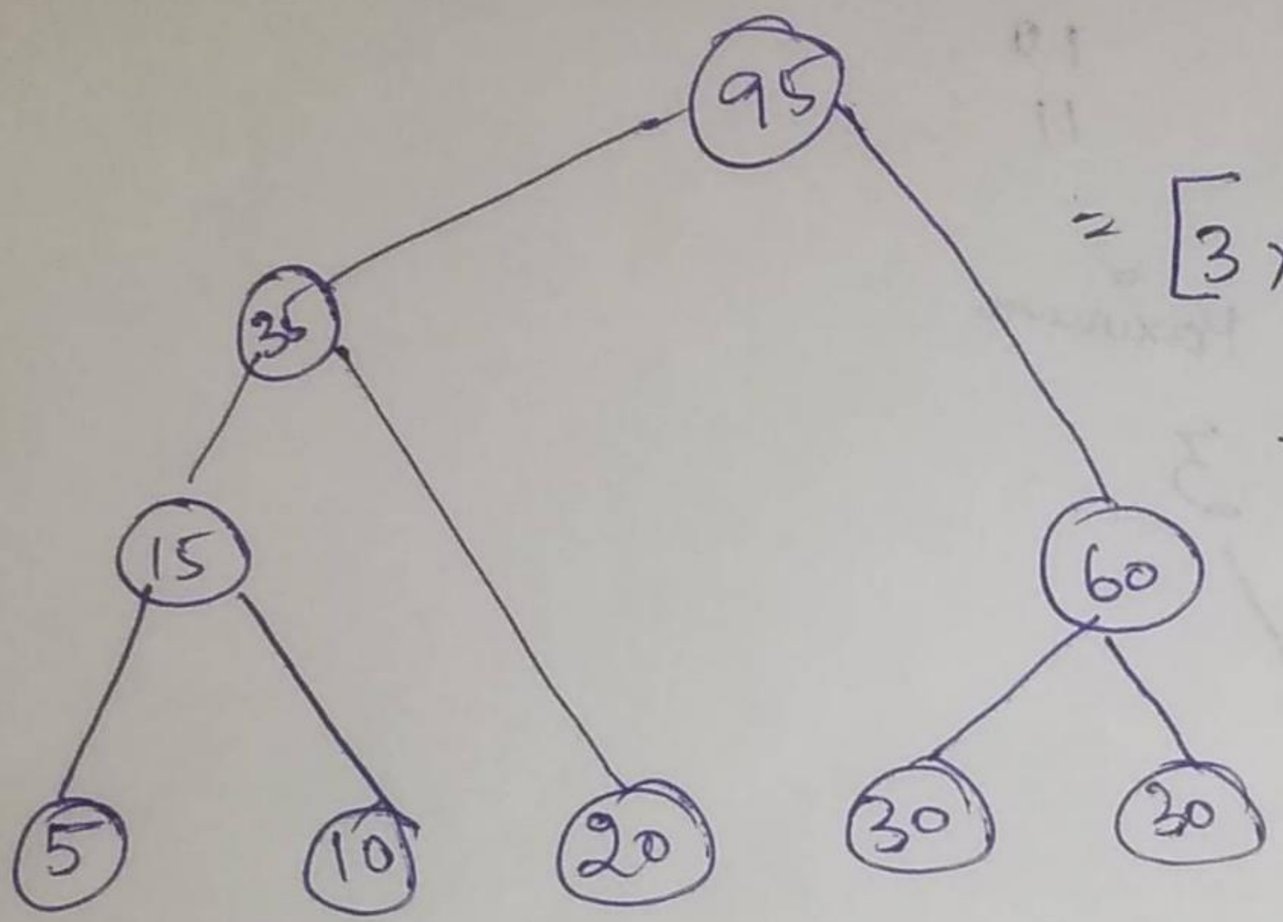
IIIrd Model



file	x_1	x_2	x_3	x_4	x_5
Size	20	30	10	5	30

Accending order.

file	x_4	x_3	x_1	x_3	x_5
size	5	10	20	30	30



$= [3 \times 5 + 3 \times 10 + 2 \times 20 + 30 \times 2 + 30 \times 2]$

$= 205$ //

Answer:

$= 15 + 35 + 95 + 60 = 205$

Algorithm :

5. Optimal Storage on tapes:

There are n programs that are to be stored on a computer tape of length l. Associate with each Program, i is a length l_i, $1 \le i \le n$. Correctly, all Programs can be stored on the tapes if and only if the sum of the lengths of the programs is at most l.

Example:

Let $n=3$ and $(l_1, l_2, l_3) = (5, 10, 3)$ there are $n! = 6$ possible orderings. These orderings and their respective d values are

Ordering	d(I)
1, 2, 3	5+5+10+5+10+3 = 38
1, 3, 2	5+5+3+5+3+10 = 31
2, 1, 3	10+10+5+10+5+3 = 43
2, 3, 1	10+10+3+10+3+5 = 41
3, 2, 1	3+3+10+3+10+5 = 34

Eg: {12, 34, 56, 73, 24, 11, 34, 56, 78, 91, 34, 4} on three tapes

Store files in non-decreasing length.

{11, 12, 24, 34, 34, 45, 56, 56, 78, 78, 91}

tape 1 →	11,	34	45	73
tape 2 →	12	34	56	78
tape 3 →	24	34	56	91

Algorithm:

```
// n is the no of programs and m is the no of tapes
{
  j:=0
for i:=1 to n do
{
while("opend program",i, to permutation for tape
j=(j+1) mod m
}
}
```

6. Knapsack Problem:

To apply the greedy method to some problem Consider knapsack Problem. we are given n object and a knapsack has bag, object i has a wi and the knapsack has a capacity m. A profit of PiWi is earned.

Problem:

Consider the following instant of the knapsack problem

$n=3$, $m=20$ $(P_1, P_2, P_3) = (24, 25, 15)$, $(W_1, W_2, W_3) = (18, 15, 10)$

i) (P_1, P_2, P_3)

$\left(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}\right) \Rightarrow (24, 25, 15)$

$\Rightarrow \left(\frac{24}{2}, \frac{25}{3}, \frac{15}{4}\right) = (12, 8.3, 3.7)$

$\Rightarrow (24.0)$

ii) $(1, \frac{2}{15}, 0) \xrightarrow{1:1} (24, 25, 15)$

$\Rightarrow (24, \frac{2 \times 25}{15_3}, 0) = (24, 3.6) = 27.2$

iii) $(0, \frac{2}{3}, 1) \Rightarrow (24, 25, 15)$

$\Rightarrow (0, \frac{2 \times 25}{15}, 15) = (16.6, 15)$

$\Rightarrow (31.6)$

iv) $(0, 1, \frac{1}{2}) \Rightarrow (24, 25, 15)$

$\Rightarrow (0, 25, \frac{1 \times 15}{2}) \Rightarrow (25, 7.5)$

$\Rightarrow (32.5)$

W_1, W_2, W_3

i) $(\frac{1}{2}, \frac{1}{3}, \frac{1}{4}) \Rightarrow (18, 15, 10)$

$\Rightarrow (\frac{1}{2}, \frac{1}{3}, \frac{1}{4}) \Rightarrow (9, 5, 2.5)$

$\Rightarrow (16.5)$

ii) $(1, \frac{2}{15}, 0) \Rightarrow (18, 15, 10)$

$\Rightarrow (18, \frac{2 \times 15}{15}, 0) = (18, 2)$

$\Rightarrow (20)$

iii) $(0, \frac{2}{3}, 1) \Rightarrow (18, 15, 10)$

$\Rightarrow (0, \frac{2 \times 15}{3}, 10) \Rightarrow (0, 10, 10)$

$\Rightarrow (20)$

iv) $(0, 1, \frac{1}{2}) \Rightarrow (18, 15, 10)$

$\Rightarrow (0, 15, \frac{15}{3}) \Rightarrow (15, 5)$

$\Rightarrow (20)$

TABLE:

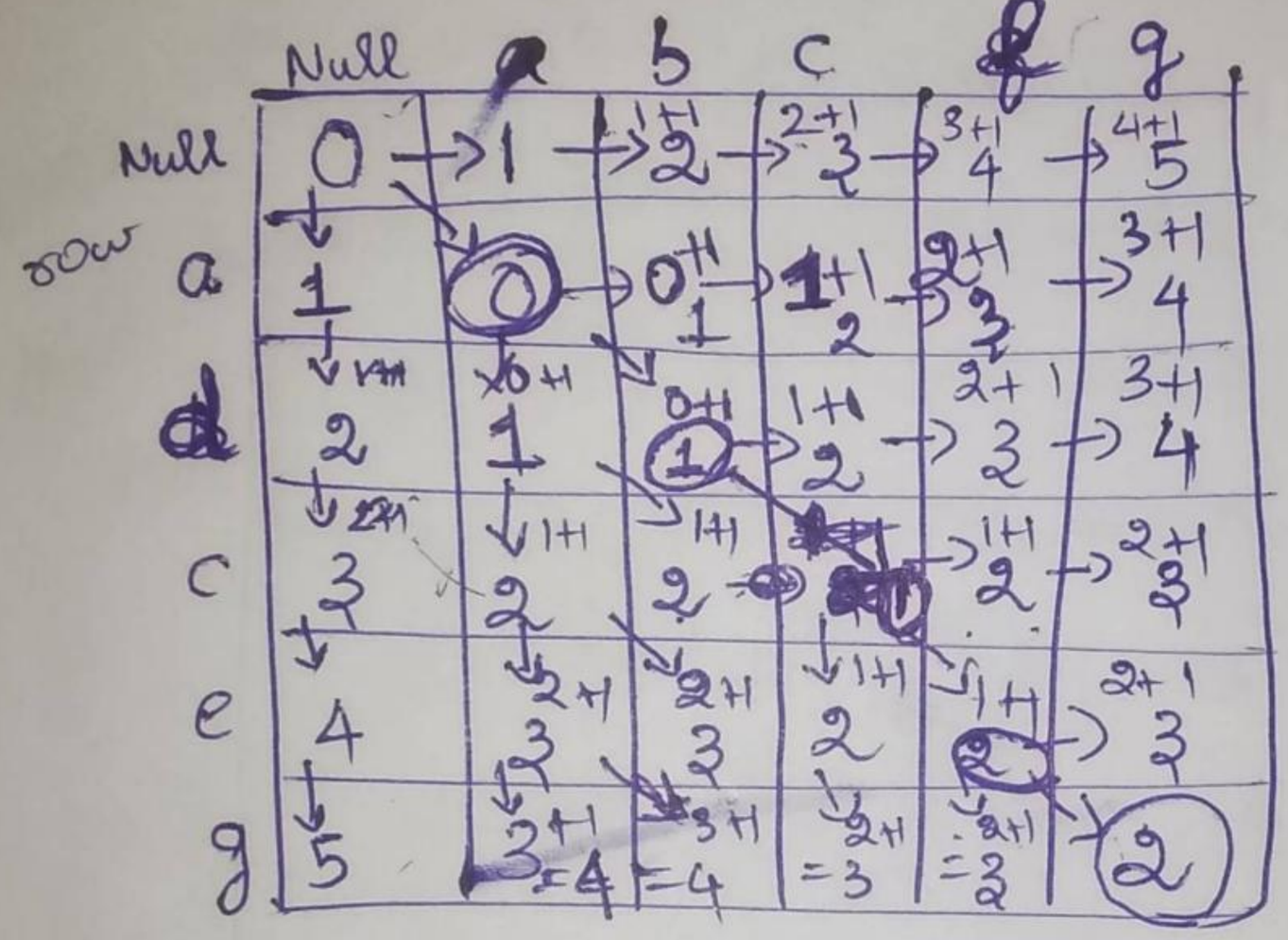
S.NO	$x_1 x_2 x_3$	$\sum w_i x_i$	$\sum p_i x_i$
1.	$(\frac{1}{2}, \frac{1}{3}, \frac{1}{4})$	16.5	24.0
2.	$(1, \frac{2}{15}, 0)$	20	27.2
3.	$(0, \frac{2}{3}, 1)$	20	31.6
4.	$(0, 1, \frac{1}{2})$	20	32.5

Algorithm:

```
Void Greedy method (float m; int n)
{
for (int i=1 ; i<=n ; i++)      x[i] =0.0;
float v=m;
for (i=1 ; i<=n; i++)
{
if (W[i]>v) break;
  x[i]=1-0;
ux  = W[i]
r   }
if (i<=n) x[i] - y[i];
}
```

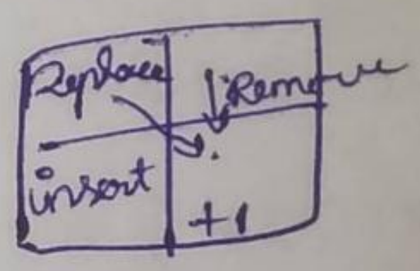
Unit-4

String Editing



↓ Remove
→ insert
↘ Replace.

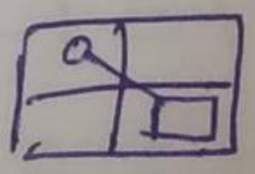
1. If $r \neq c$



2. If $r == c$

Just copy the diagram

element



2

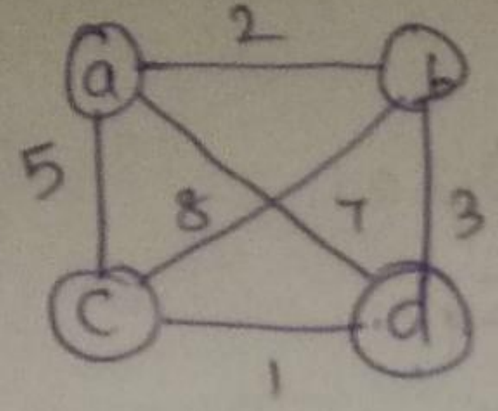
→ Replace.

a d c e g

a b c f g

```
for (i=0; i < len (string )+1 ; i++)
 {
  for (j=0 ; j< len( strg) +1 ; j++)
   {
    if (i==0 && j==0)
```

1)

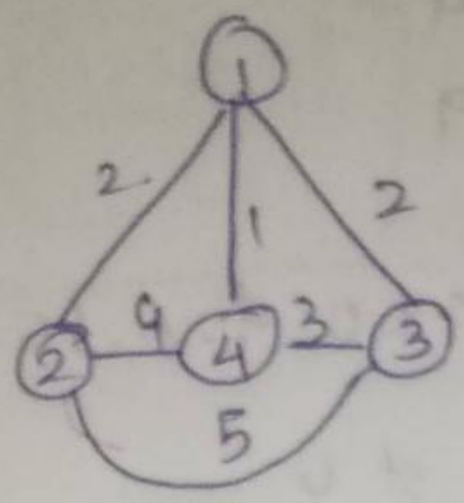


all visit vertices

But

Start vertices = end vertices

2)



1) Solution:

minimum value

$a \to b \to c \to d \to a = 2+8+1+7 = 18$

Optimal sol

$a \to b \to d \to c \to a = 2+3+1+5 = 11$

$a \to c \to b \to d \to a = 5+8+3+7 = 23$

$a \to c \to d \to b \to a = 5+1+3+2 = 11$

$a \to d \to c \to b \to a = 7+1+8+2 = 18$

$a \to d \to b \to c \to a = 7+3+8+5 = 23$

2) Solution

$1 \to 2 \to 3 \to 4 \to 1 = 2+5+3+1 = 11$✓

$1 \to 2 \to 4 \to 3 \to 1 = 2+4+3+2 = 11$✓

$1 \to 3 \to 2 \to 4 \to 1 = 2+5+4+1 = 12$✗

$1 \to 3 \to 4 \to 2 \to 1 = 2+3+4+2 = 11$✓

$1 \to 4 \to 2 \to 3 \to 1 = 1+4+5+2 = 12$✗

$1 \to 4 \to 3 \to 2 \to 1 = 1+3+5+2 = 11$✓

Optimal Solution:

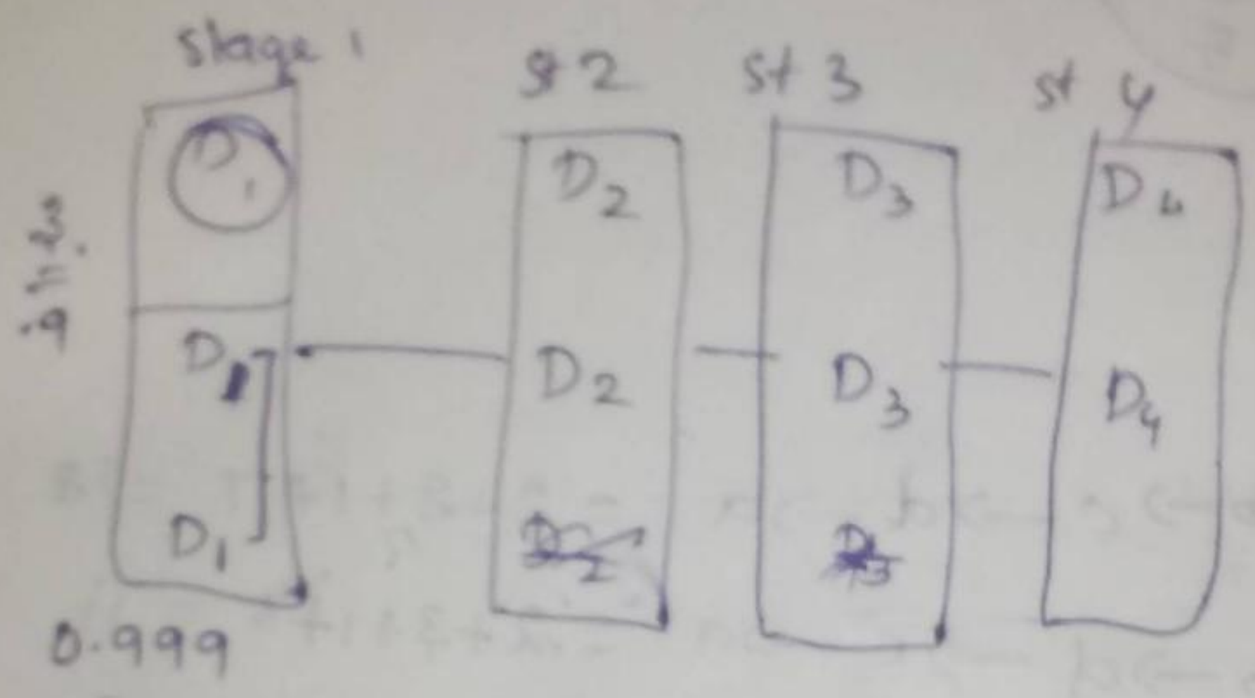
3. Reliability Design

Setup a system

$D_1 - D_2 - D_3 - D_4$

C_1	C_2	C_3	C_4
r_1	r_2	r_3	r_4
.9	.9	.9	.9

$\pi r_1 = 0.9 * 0.9 * 0.9 * 0.9 = 0.6561$



$r_1 = 0.9$

$1 - r_1 = 1 - 0.9 = 0.1$

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

$1 - (1 - r_1)^3 = \underline{0.999}$

Example Program

D_i	C_i	r_i	u_i
D_1	30	0.9	2
D_2	15	0.8	3
D_3	20	0.5	3

$C = 105$

$$\sum c_i = c_1 + c_2 + c_3$$

$$= 30 + 15 + 20 = 65$$

$$C - \sum c_i = 105 - 65 = 40$$

$$= \frac{40}{30} = 1$$

$$\mu_i = \left[\frac{C - \sum c_i}{c_i}\right] + 1 = \frac{40}{30} = 1 + 1 = \underline{2}$$

$$= \frac{40}{15} = 2 + 1 = 3$$

$$= \frac{40}{20} = 2 + 1 = 3$$

Reliability Design

(R, C)

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

$S_1^1 = \{(0.9, 30)\}$

$1 - (1 - r_1)^2 = 1 - (1 - 0.9)^2$

$= 1 - (0.1)^2$

$= 1 - 0.01$

$= 0.99.$

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

$S^1 = \{(0.9, \underline{30}), (0.99, \underline{60})\}$

Di	ci	ri	mi
D_1	30	0.9	2
D_2	15	0.8	3
D_3	20	0.5	3

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

$((0.9 \times 0.8), 30 + 15)$ $(0.99 \times 0.8, 60 + 15)$

$1 - (1 - r_1)^2$

$1 - (1 - 0.8)^2$

$1 - (0.2)^2$

$1 - 0.04$

$= 0.96$

$S_2^2 = \{(0.864, 60), (0.6504, \underline{90})\}$

$30 + 60$

$30 + 30$

$1 - (1 - r_1)^3$

$1 - (1.08)^3$

$1 - (0.2)^3$

$1 - 0.008$

$= 0.992$ 45

$S_3^2 = \{(0.8928, 75), (\quad, 105)\}$

$S^2 = \{(0.72, 45), (0.792, 75), (0.80, 60), (0.8928, 75)\}$

30.

Reliability Design

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

$s_1^1 = \{(0.9, 30)\}$ 2

D_1 $s_2^1 = \{(0.99, 60)\}$

$s^1 = \{(0.9, 30), (0.99, 60)\}$
1

D_2 $s_1^2 = \{(0.72, 45), (0.792, 75)\}$

$s_2^2 = \{(0.864, 60), (\text{---}, 90)\}$

$s_3^2 = \{(0.8928, 75), (\text{---}, 105)\}$

is increase Reliability and cost.

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

Higher cost is removed.

D_3 $s^2 = \{(0.72, 45), (0.792, 75), (0.864, 60), (0.8928, 75)\}$

$s_1^3 = \{(0.36, 65), (0.432, 80), (0.4464, 95)\}$
45+20, 60+20, 75+20

$s_2^3 = \{(0.54, 85), (0.648, 100), (\text{---}, 115)\}$

$s_3^3 = \{(0.63, 105), (\text{---}, 120), (\text{---}, 135)\}$

$r_3 = 0.5$

$1-(1-r_3)^3$

$1-(1-0.5)^3$

$1-(0.5)^3$

$1-0.125$

0.875

$r_3 = 0.5$

$1-(1-r_3)^2$

$1-(1-0.5)^2$

$1-(0.5)^2$

$1-0.25$

0.75 40

$s^3 = \{(0.36, 65), (0.432, 80), (0.4464, 95), (0.54, 85), (0.648, 100), (0.63, 105), (0.648, 100)\}$

$S^3 = \{(0, 36, 65), (0.432, 80), (0, 44, 95),$

$(0.54, 85), (0.\cancel{6}\cancel{4}8, 105), (0.648, 100)\}$ max

Maximum.

0.648

100

Ans:

0.648

100

D_1	D_2	D_3
1	2	2

$\bar{\chi}\ \gamma_1 \quad 0.648$

$\Sigma c_i \quad = 100$

Unit-5

1. Binary Tree.

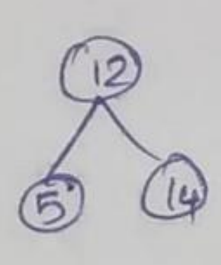
Tree - Non Linear data structure

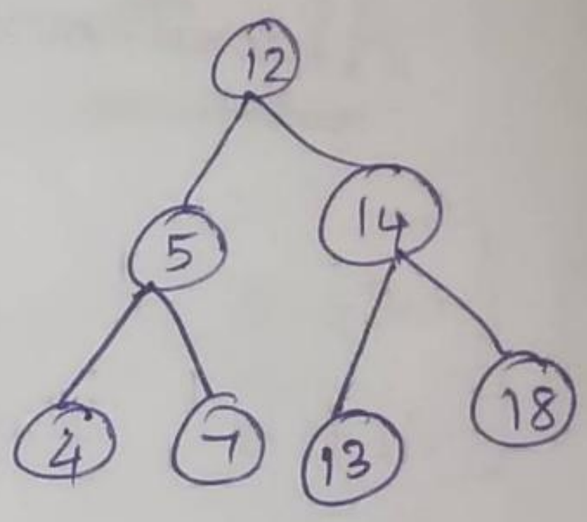
Depth first Traversals.

Preorder - Root, left, Right.
In order - Left, Root, Right
Post order - Left, Right, Root.

Preorder

Root, left, Right



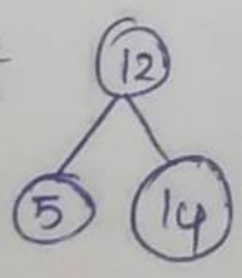


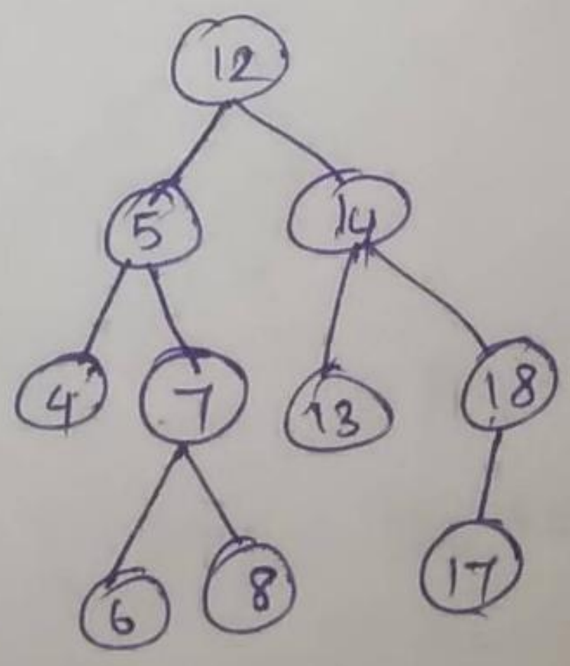
Order: 12, 5, 14.

Order: 12, 5, 4, 7, 14, 13, 18

Inorder

Left, Root, Right



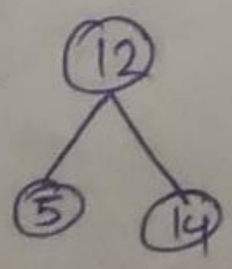


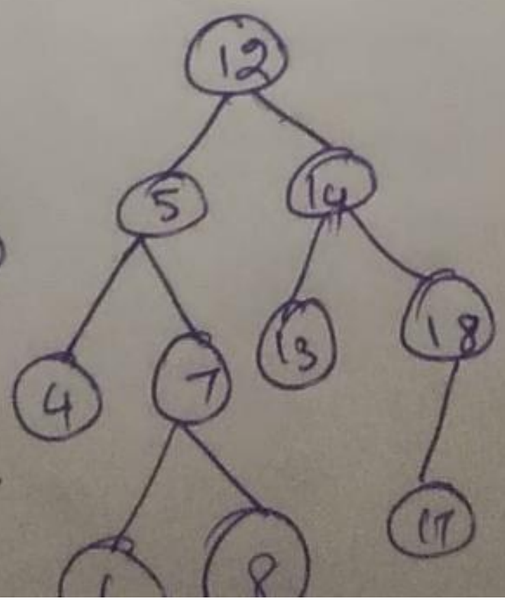
Order: 5, 12, 14

Order: 4, 6, 7, 8, 5, 12, 13, 14, 7, 18

Post order:

Left, Right, Root.





Order: 5, 14, 12.

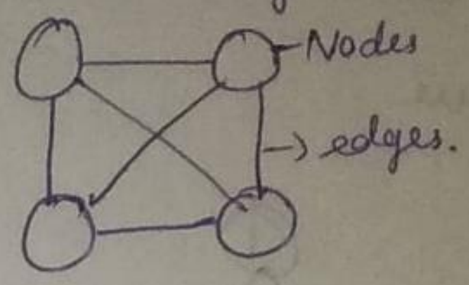
~~Order: 6, 8, 7, 4, 5, 17, 18, 13, 14, 12.~~

2. Graphs :-

1. What is a graph?

A graph is a collection of nodes called vertices and the connections between them called edges.

$G = \{V, E\}$



Complete Graph.

n nodes & $n(n-1)/2$ edges.

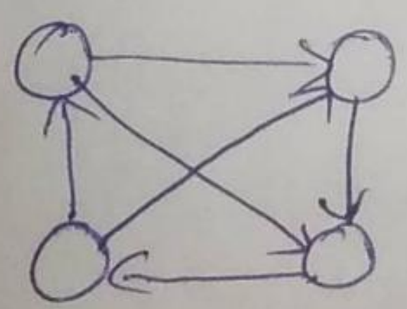
2. Basic Technology

* Length - no. of edges in path.
* Cost - sum of weights on each edge in Path
* cycle - Path starts and finishers in same node.
* Acyclic - contains no cycle.

3. Graph types

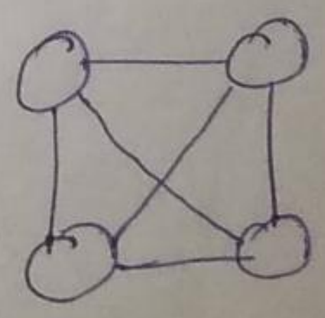
Directed types

* Vertices have directed edges.



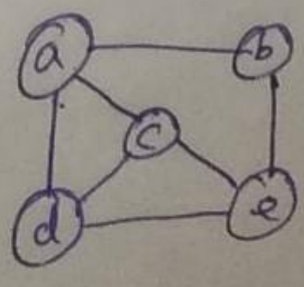
Undirected Graphs.

* Vertices don't have directed edges.



4. Representation of Graph

* Adjacency Matrix



	a	b	c	d	e
a	F	T	T	T	F
b	T	F	F	F	T
c	T	F	F	T	T
d	T	F	T	F	T
e	F	T	T	T	F

* Adjacency List.

- Sequence of nodes adjacent to node v

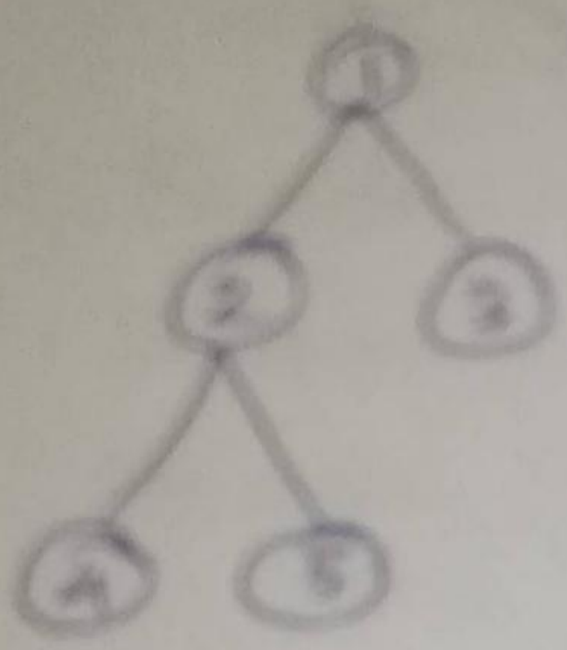
5. Graph Traversals

* BFS

→ Visit all siblings before their descendants

→ Based on FIFO queue

Queue



Algorithm

* start from node 1
* visit all neighbours of node 1
* Now from left to right
* Then visit of all their neighbour, if not already visited
* Continue until all nodes visited.

* DFS

→ Visit all descendants as deep as possible and then visit the siblings

→ Based on LIFO queue Stack

Algorithm

* Given an starting vertex
* Pick an adjacent vertex, visit it,
→ then visit one of it adjacent vertices
→
→ until impossible, then backtrack, visit another.

3. Back tracking :-

It is used to solve problem.

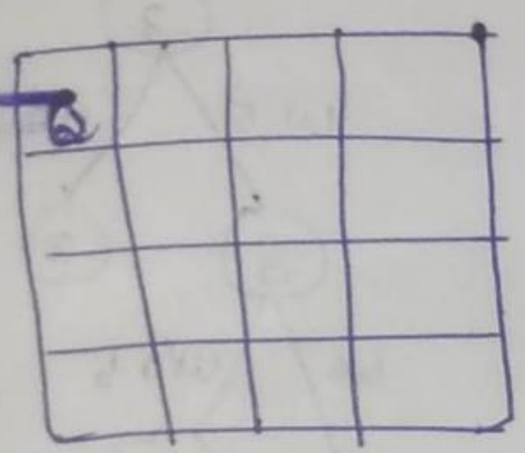
I st method.

N- Queen problem.

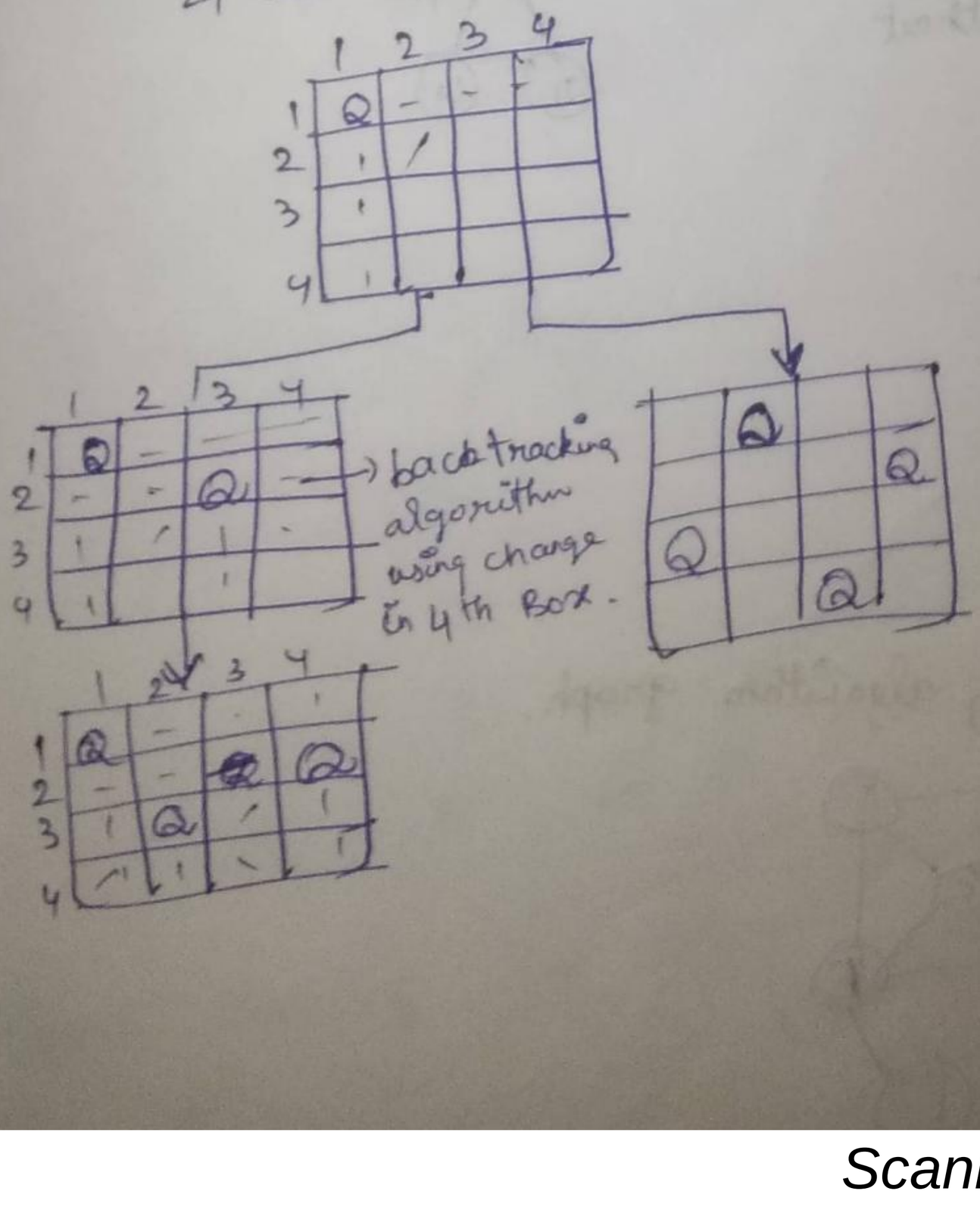
↳ 4 Queen

↳ 8 Queen.

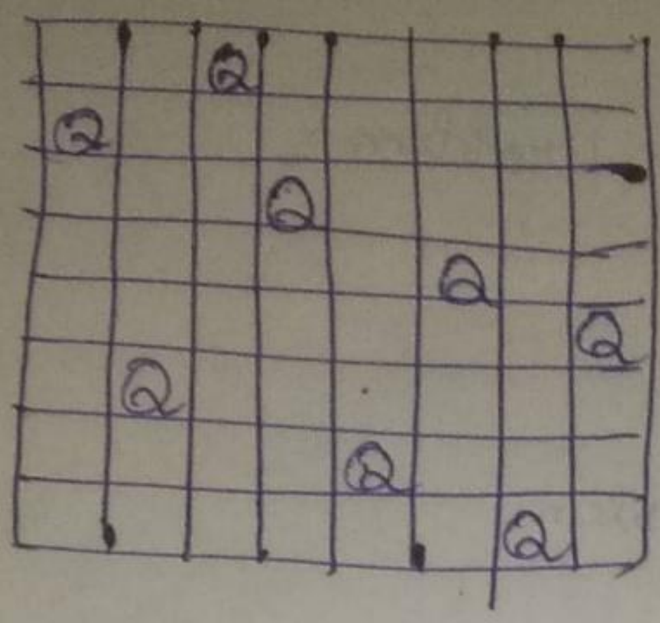
4 Queen.



4 Queen :



8 Queen



2. Subset

$S = \{3, 5, 6, 7\}$.

Condition
1. with
2. without

$S = S_1, * S_2, * S_n$

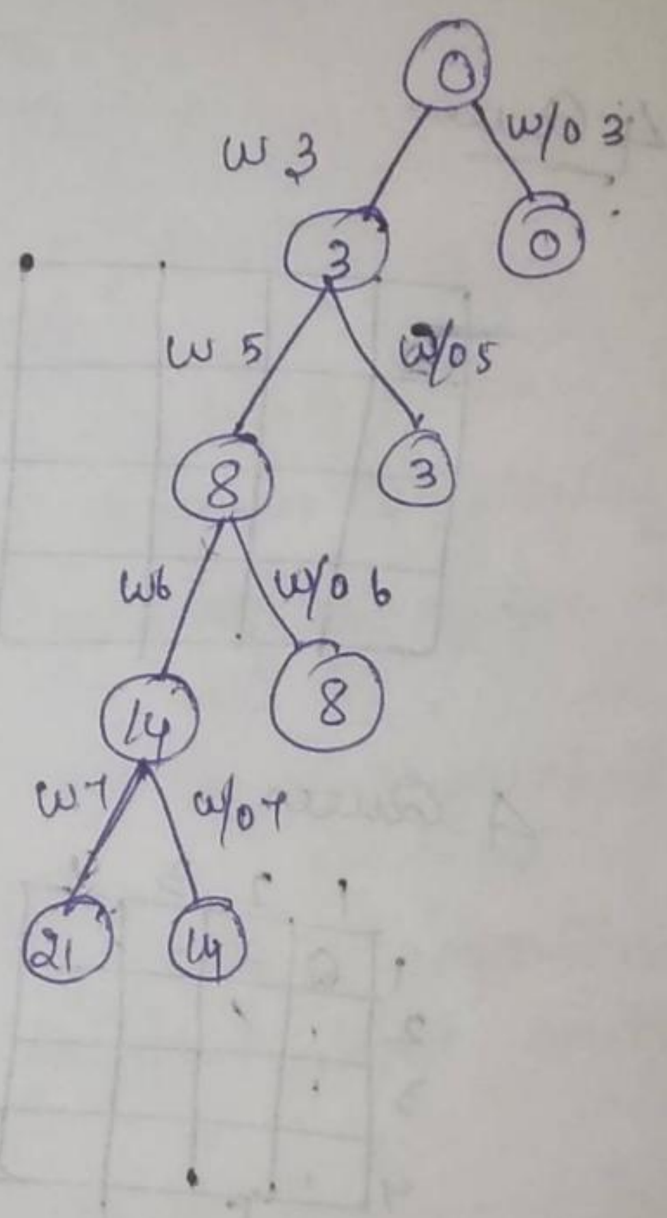
initial node = 0.

Right side = with

Left side = without

with = add.

without = No add
Node value.

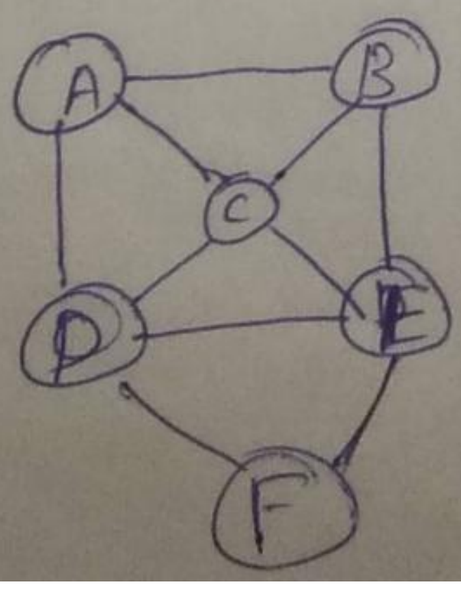


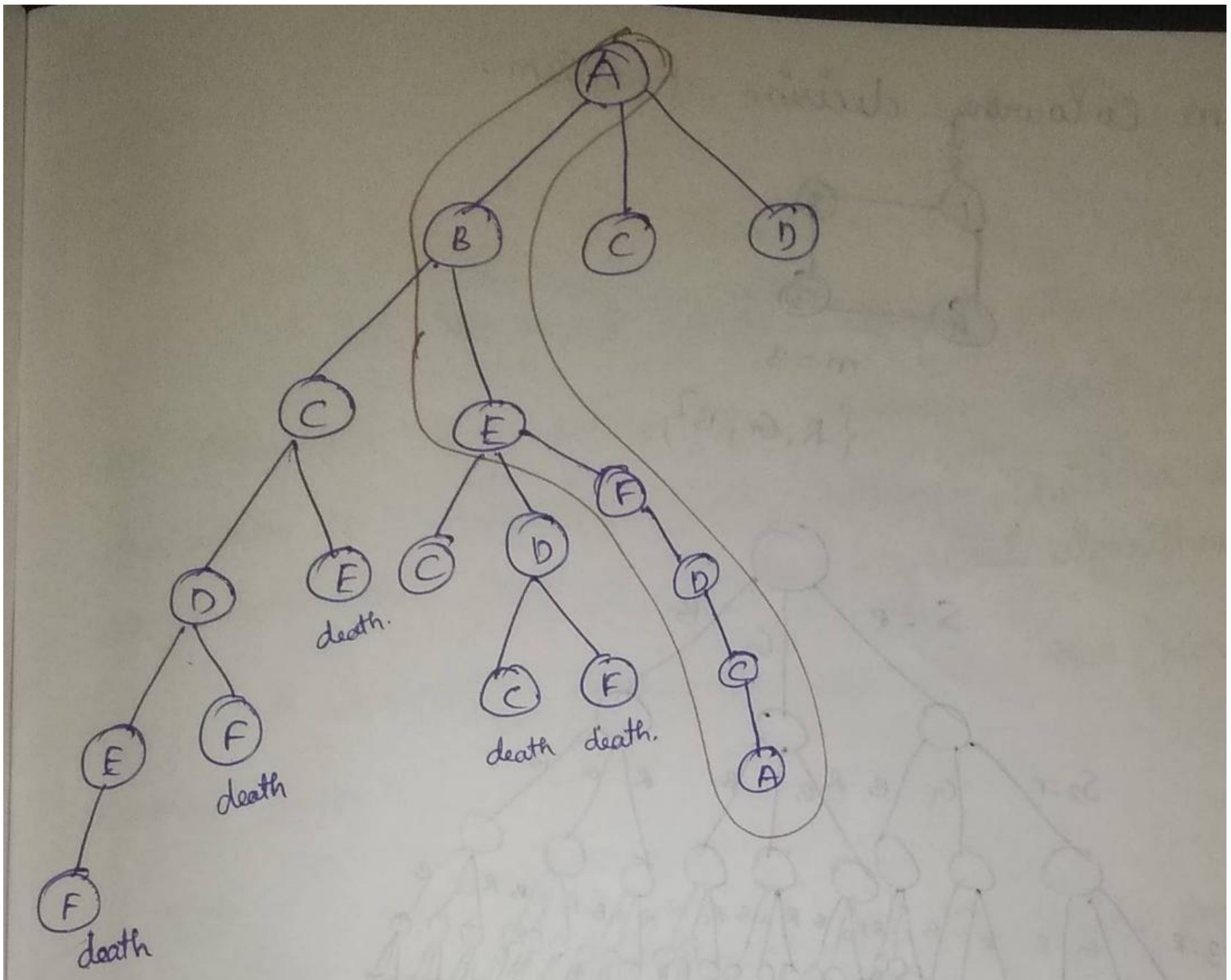
Ex: Node = 0

with = 0+3 = 3

without = 3.

3. Back Tracking algorithm graph.



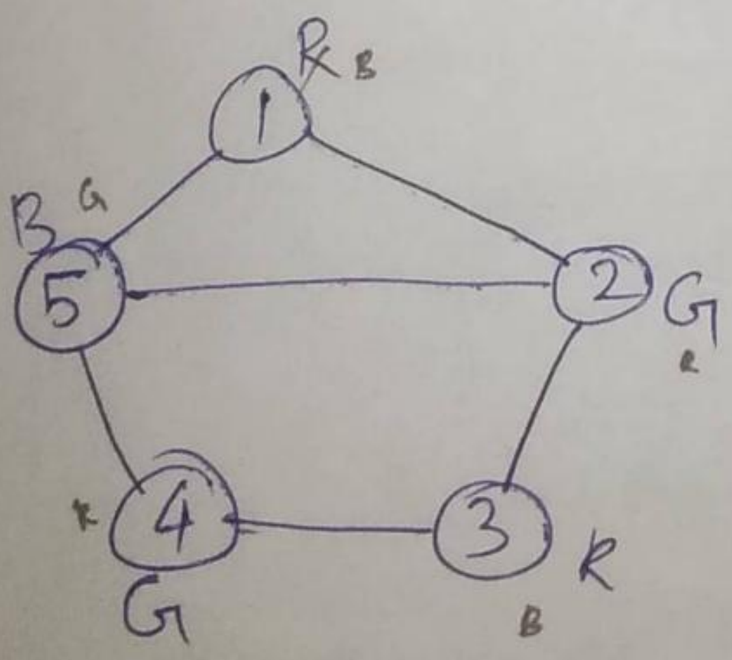


4. Graph Colouring Problem.

Back Tracking

R, G, B

m=3

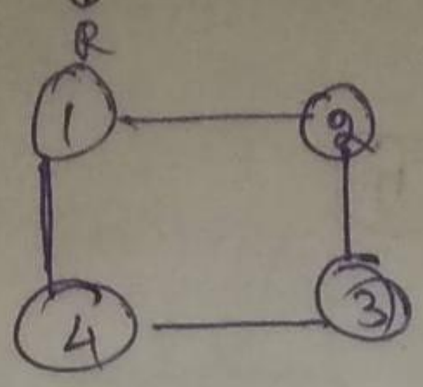


1	2	3	4	5
R	G	R	G	B

Back tracking using colour.

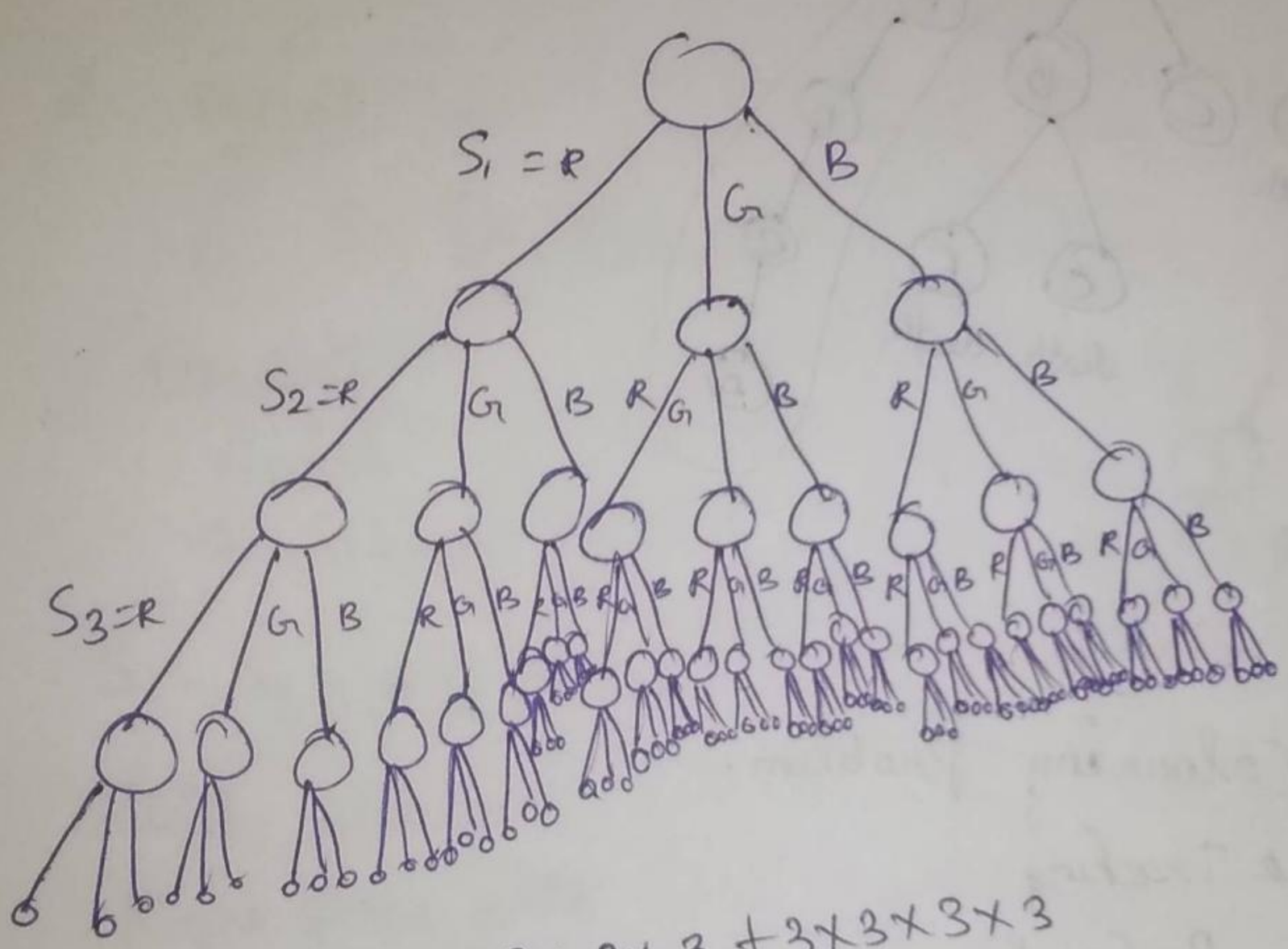
R, G, B

m Colouring decision Problem.



$m = 3$

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



$1 + 3 + 3\times3 + 3\times3\times3 + 3\times3\times3\times3$

$1 + 3 + 3^2 + 3^3 + 3^4 = \frac{3-1}{3-1}$