No.

Date

* INTRODUCTION
* COMPLEXITY & ASYMPATIC NOTATION

No.

Date

ALGOS : GREEK ROOT 'PAIN'

ALGORITHM : A FINITE SET OF INSTRUCTIONS THAT, if FOLLOWED, ACCOMPLISH A PARTICULAR TASK.

A WELL DEFINED PROCEDURE THAT TRANSFERS AN INPUT TO AN OUTPUT.

1. INPUT : has zero or more inputs.
2. OUTPUT : has at least one output.
3. DEFINITENESS : each step is precisely described.
4. FINITENESS : must terminate after a finite number of steps
5. EFFECTIVNESS : each operation should be sufficiently basic such that they can be done in finite amount of time. using pencil & paper.

MODEL OF COMPUTATION : RUN TIME & ~~TA~~ SPACE

RUN TIME : RAM consists infinity array of cells, each cell can store at most one data. each memory can access in unit time, instructions are executed sequentially, all basic instructions take unit time :
① LOAD/STORE/ ② ARITHMETIC (+ - * /) ③ LOGIC (>, <)

ALGOS : greek root, PAIN.

RAM NOT REALISTIC.
FINITE MEMORY, NOT ALL MEMORY TAKE SAME TIME, NOT ALL OPERATIONS TAKE SAME TIME, PIPELINING.

9th century, Parsian Mathematician, Musa-al-Khwarizm

algebra → algorism

ASYMPTOTIC NOTATIONS:

① BIG OH FUNCTION $O(n)$: $f(n) = O(g(n))$

iff there exists positive constants c & n_0 such that $f(n) \leq c * g(n)$, for all n, $n \geq n_0$

$3n+2 = O(n)$ as $3n+2 \leq 4n$; $n \geq 2$
$3n+3 \leq 4n$, $n \geq 3$

$10n^2 + 4n + 2 = O(n^2)$
$\Rightarrow 10n^2 + 4n + 2 \leq 11n^2$, $n \geq 5$

$1000n^2 + 100n - 6 = O(n^2)$
$\Rightarrow 1000n^2 + 100n - 6 \leq 1001n^2$, $n \geq 100$

$6 * 2^n + n^2 = O(2^n)$
$\Rightarrow 6 * 2^n + n^2 \leq 7 * 2^n$, $n \geq 4$

No.

Date

(II) BIG OMEGA FUNCTION $\Omega(0)$: $f(n) = \Omega(g(n))$

iff there exists positive constants c & n_0 such that $f(n) \geq c * g(n)$. for all n, $n \geq n_0$ & $n_0 > 0$

$f(n) = 3n + 2 \geq 3n \quad n \geq 1$

$10n^2 + 4n + 2 \geq n^2, \quad n \geq 1$

$6 * 2^n + n^2 \geq n^2, \; n \geq 1$

No.

Date

(III) BIG THETA FUNCTION $\Theta(g(n))$:

iff there exists positive constants c_1, c_2 & n_0 such that, $c_1 * g(n) \leq f(n) \leq c_2 * g(n)$, all n, $n \geq n_0$

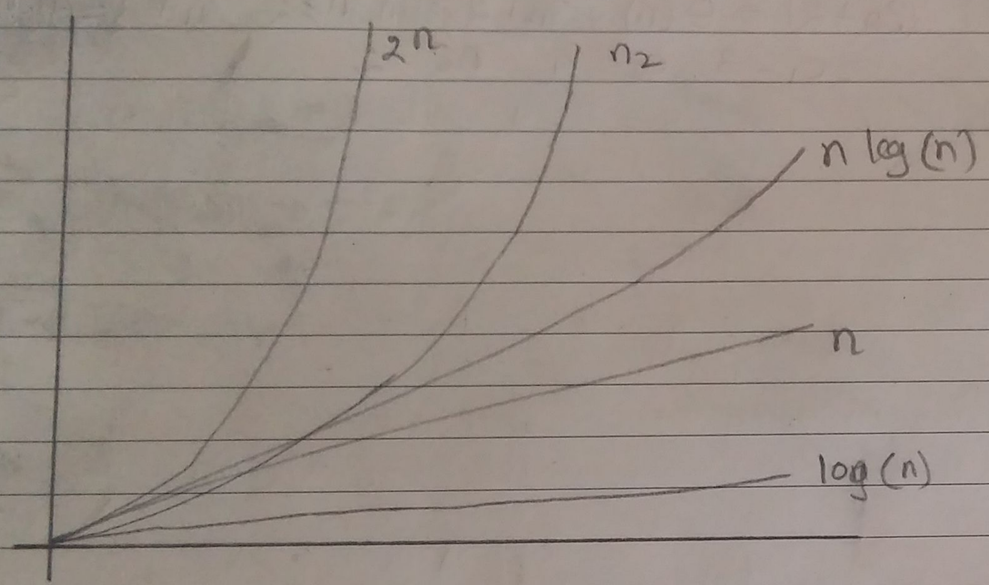
$(3n+2) = \Theta(n)$; $3n+2 \geq 3n, n \geq 2$, $3n+2 \leq 4, n \geq 2$

$c_1 = 3$, $c_2 = 4$, $n_0 = 2$

SPACE - TIME COMPLEXITY:

SPACE COMPLEXITY: the amount of memory it needs to run to completion.

TIME COMPLEXITY: the amount of computer time it needs to run to completion.



PLOT of FUNCTIONS VALUES

COMPUTE x^n:

NAIVE ALGORITHM:

```
power = x;

for(i=1; i≤n; i++)
{
    power = power * x;
}
```

x^n, when n is power of two

```
power = x;

for(i=1; i ≤ log2 n; i++)
{
    power = power * power;
}
```

x^n, for any value of n

```
m=n, power=1, z=x
while (m>0)
{
    while (m%2==0)
    {
        m = m/2;
        z = z*z;
    }
    m=m-1; power = power * z;
    return power;
}
```

SEARCH ALGORITHM:

! LINEAR SEARCH:

complexity : O(n)

```
linear search (A[], n, data)

for (i=1; i<=n; i++)
{
    if (A[i]==data)
    {
        index = i;
        break;
    }
}
```

1	2	3	4	5	6	7	8

A[] = {4, 6, 2, 3, 8, 1, 5, 2} data = 5

INDEX = 7

!! BINARY SEARCH:

complexity ($\log_2 n$)
sorted data

```
binary search (A[], n, data)

beg = 1, end = n;

while (beg ≤ end)

    mid = (beg + end) / 2;

    if data > A[mid]
        beg = mid + 1;
    else
        end = mid;

    if data==A[mid]
        location = mid
    else
        location = 0
```

1	1	2	3	4	5	6	8	6
S			M				E	
			S		M		E	
					S	M	E	

!!! INDEXING: // NEGATIVE DATA complexity O(1)

```
hashing (A[], n, data, index[])

for(i=1; i≤n; i++)
{
    index [A[i]] = 1;
}

if (index [data] == 1)
{   print("Yes");
}
```

A[ ]	1	1	2	3	4	5	6	8
index[ ]	1	2	3	4	5	6	7	8
	1	1	1	1	1	1	0	1

data : 6

! INSERTION SORT :

data: 4 3 1 5 6 2 7 8

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

!! BUBBLE SORT :

2 4 3 1 4 6 2
2 4 3 1 4 6 2
2 4 3 1 4 6 2
1 4 3 2 4 6 2
1 4 3 2 4 6 2
1 4 3 2 4 6 2

1 4 3 2 4 6 2
1 3 4 2 4 6 2
1 2 4 3 4 6 2
1 2 4 3 4 6 2
1 2 4 3 4 6 2

1 2 4 3 4 6 2
1 2 3 4 4 6 2
1 2 3 4 4 6 2
1 2 3 4 4 6 2

1 2 2 4 4 6 3
1 2 2 4 4 6 3
1 2 2 4 4 6 3

1 2 2 3 4 6 4
1 2 2 3 4 6 4

1 2 2 3 4 6 6

1 2 2 3 4 4 6

A2M SUPREME

!!! MERGE SORT :

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

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

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

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

IV QUICK SORT:

$\boxed{2}$ 8 7 1 3 5 6 4

$\boxed{2}$ 1 7 8 3 5 6 4

1 $\boxed{2}$ 7 8 3 5 6 4

1 2 $\boxed{7}$ 8 3 5 6 4

1 2 $\boxed{7}$ 4 3 5 6 8

1 2 6 4 3 5 $\boxed{7}$ 8

1 2 $\boxed{6}$ 4 3 5 7 8

1 2 5 4 3 $\boxed{6}$ 7 8

1 2 $\boxed{5}$ 4 3 6 7 8

1 2 3 4 $\boxed{5}$ 6 7 8

1 2 $\boxed{3}$ 4 5 6 7 8

1 2 3 4 5 6 7 8

8	3	5	6	10	17	1	2	18	25
8	3	5	6	2	17	1	10	18	25
8	3	5	6	2	1	17	10	18	25
1	3	5	6	2	8	17	10	18	25

1	3	5	6	2	8	17	10	18	25
1	3	6	6	2	8	17	10	18	25
1	3	5	6	2	8	10	17	18	25
1	3	2	6	5	8	10	17	18	25
1	2	3	6	5	8	10	17	18	25
1	2	3	6	5	8	10	17	18	25
1	2	3	5	6	8	10	17	18	25

GREEDY ALGORITHM:

→ optimal memory

→ faster

→ does not always work

ACTIVITY SELECTION PROBLEM:

INPUT: A LIST OF ACTIVITY WITH START TIME & END TIME.

OUTPUT: MAXIMUM NUMBER OF SELECTED ACTIVITY, NOT CONFLICTING WITH EACH OTHER

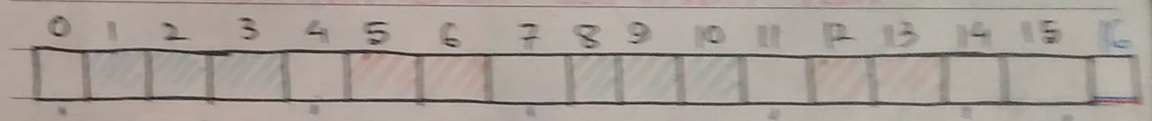
ACTIVITY SELECTION PROBLEM:

Activity:	1	2	3	4	5	6	7	8	9	10	11
Start :	0	1	2	3	3	5	5	6	8	8	12
End :	6	4	13	5	8	7	9	10	11	12	14

SOLUTION:

SORT BY END TIME:

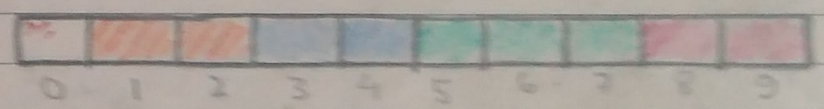
Activity:	2	4	1	6	5	7	8	9	10	3	11
Start:	1	3	0	5	3	5	6	8	8	2	12
End:	4	5	6	7	8	9	10	11	12	13	14



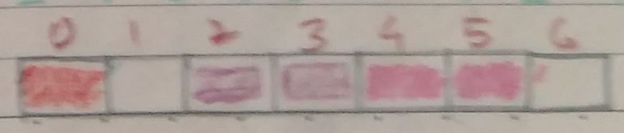
ACTIVITY:	1	2	3	4	5	6
START:	0	1	3	5	5	8
END:	6	2	4	7	9	9

ACTIVITIES

~~ACTIVITY~~:	2	3	1	4	6	5
START:	1	3	0	5	8	5
END:	2	4	6	7	9	9



ACTIVITY:									
START:	0	0	1	2	4	0	0	2	1	4
END:	2	1	5	4	6	1	2	4	5	6



GREEDY KNAPSCAK PROBLEM:

INPUT: A LIST OF N ITEMS WITH PRICE & WEIGHT.
OUTPUT: MOST PROFITABLE ITEMS. THAT CAN

FRACTIONAL KNAPSCAK PROBLEM:

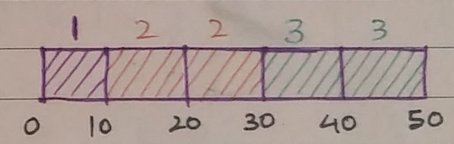
ITEM	WEIGHT	PRICE
1	10	60
2	20	100
3	30	120

50 KG

1+3 = 20+30 = 50 / 220 TK
1+2 = 10+20 = 30 / 160 "
1+3 = 10+30 = 40 / 180 "
1+2+3 = 10+20+20 = 50 / 240 "

NAME
TV 3 30
MOBILE
10
200 +60 =260
2 40 5 20
5 30 4 6
5 50 1 10
200+60+90 = 350
5×40 = 200
1×60 = 60
3×30 = 90

ITEM	WEIGHT	PRICE	PRICE/WEIGHT
1	10	60	6
2	20	100	5
3	30	120	4



PRICE: 6 × 10 = 60
+ 5 × 20 = 100
+ 4 × 20 = 80
240 TK

HUFMAN CODING:

BINARY CHARACTER CODE: FIXED LENGTH

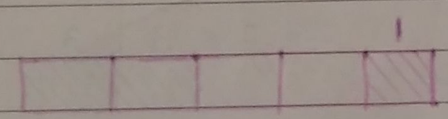
A 001	D 101
B 010	E 110
C 011	F 111

TOTAL CHARACTER: 10000
11 BITS : 30,000

How to improve?

VARIABLE LENGTH, HUFMAN CODE

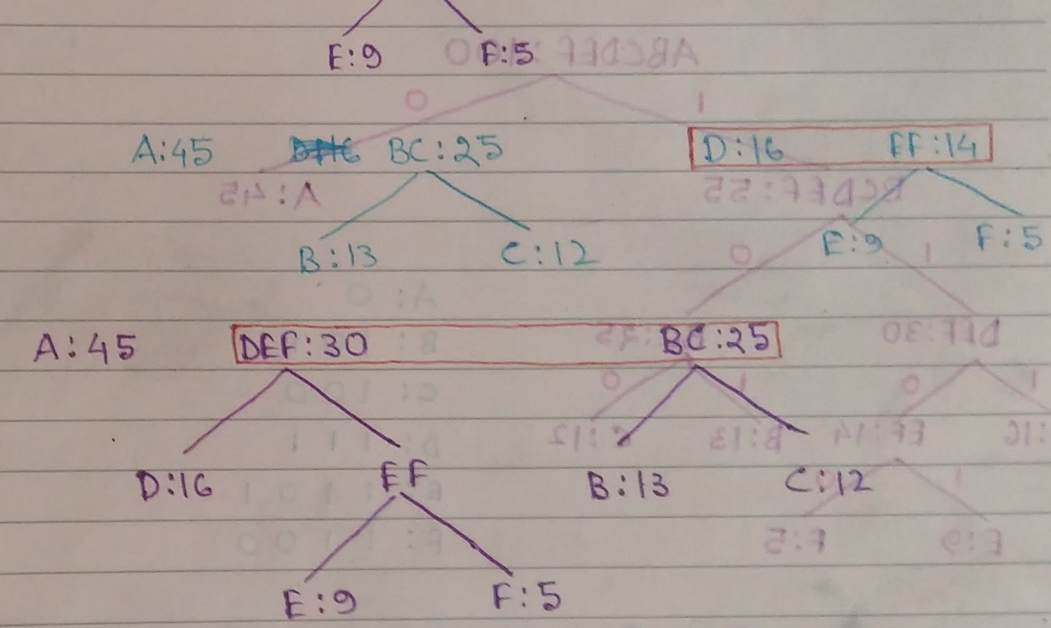
PREFIX CODE: NO CODEWORD IS ALSO A PREFIX OF SOME OTHER CODEWORD.

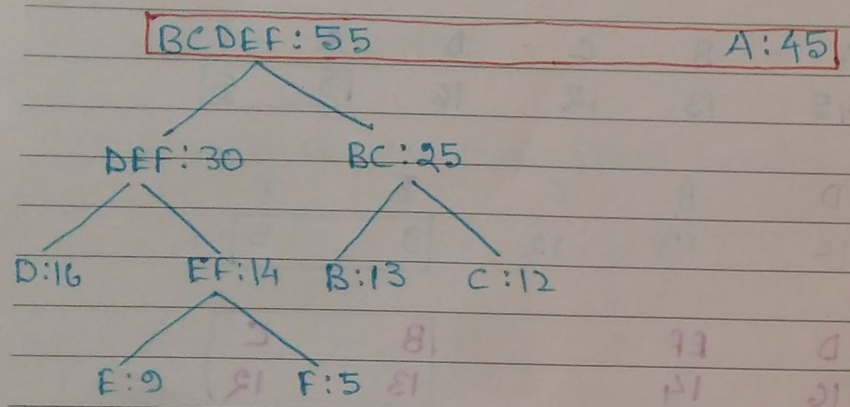


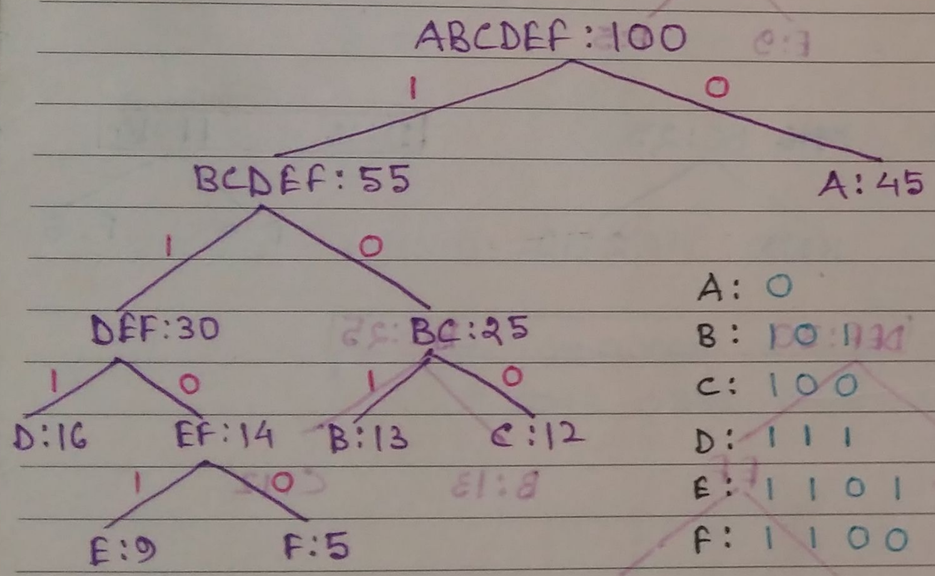
LETTER:	A	B	C	D	E	F
FREQUENCY:	45	13	12	16	9	5

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

A	D	EF	B	C
45	16	14	13	12



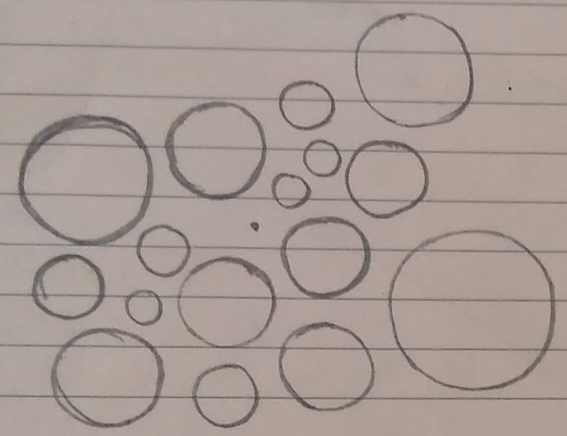




A: 0
B: 1 0 1
C: 1 0 0
D: 1 1 1
E: 1 1 0 1
F: 1 1 0 0

LETTER :	A	B	C	D	E	F	
CHARACTER :	1	3	3	3	4	4	
FREQUENCY :	45	13	12	16	9	5	= 224
FIXED :	3	3	3	3	3	3	= 300

A2M SUPREME



DYNAMIC ALGORITHM:

FIBONACCI NUMBER:

0 1 1 2 3 5 8 13 21 34

$F_0 = 1, \quad F_1 = 1; \quad F_n = F_{n-1} + F_{n-2}$

```
fibo (n)
{   F(0) = 0, F(1) = 1, F(2) = 1;
    for(i = 3; i ≤ n; i++)
    {
        F(i) = F(i-2) + F(i-1);
    }
}
```

LONGEST COMMON SUB-SEQUENCE :

↖ ↑ ← ↖

		Y	B	D	C	A	B	A
	X	0	0	0	0	0	0	0
	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
	B	0	↖ 1	← 1	← 1	← 1	↖ 2	← 2
	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4

```
print_lcs (int i, int j)
{ if(i==0 || j==0) return ;
  if (b[i][j]=='c'){
        print_lcs (i-1, j-1);
        printf (" %c", x[i-1]); }
  else if (b[i][j]=='u')
        print_lcs(i-1,j);
  else      print_lcs(i,j-1);
}
void lcs(){
    m=strlen(x); n=strlen(y);
    for (i=0; i<=m; i++) c[i][0]=0;
    for (i=0; i<=n; i++) c[0][i]=0; }
for(i=1; i<=m; i++){
  for(j=1; j<=n; j++){
    if (x[i-1]==y[j-1]){
        c[i][j]=c[i-1][j-1]+1;
        b[i][j]='c';
    }
    else if(c[i-1][j]>=c[i][j-1]){
        c[i][j]=c[i-1][j];
        b[i][j]='u';
    }
    else{
        c[i][j]=c[i][j-1];
        b[i][j]='l';
    }
void main()
{ gets(x); gets(y);
  lcs();
}
```

A2M SUPREME

$$M[x,y] = \max \begin{cases} m[x-1, y]+0 \\ m[x, y-1]+0 \\ m[x-1, y-1]+1 \ ; \text{ if matched} \end{cases}$$

	y_i	C	B	A	C	D	B	B	A	C	B	A
x_j	0	0	0	0	0	0	0	0	0	0	0	0
C	0	↖ 1	← 1	← 1	↖ 1	← 1	← 1	← 1	← 1	↖ 1	← 1	← 1
B	0	↑ 1	↖ 2	← 2	← 2	← 2	↖ 2	↖ 2	↖ 2	← 2	↖ 2	← 2
A	0	↑ 1	↑ 2	↖ 3	← 3	← 3	← 3	← 3	↖ 3	← 3	← 3	↖ 3
B	0	↑ 1	↖ 2	↑ 3	← 3	← 3	↖ 4	↖ 4	← 4	← 4	↖ 4	← 4
A	0	↑ 1	↑ 2	↖ 3	← 3	← 3	↑ 4	↑ 4	↖ 5	← 5	← 5	← 5
C	0	↑ 1	↑ 2	↑ 3	↖ 4	← 4	← 4	← 4	↑ 5	↖ 6	← 6	← 6
D	0	↑ 1	↑ 2	↑ 3	↑ 4	↖ 5	← 5	← 5	← 5	↑ 6	↑ 6	↑ 6
B	0	↑ 1	↖ 2	↑ 3	↑ 4	↑ 5	↖ 6	↖ 6	← 6	↑ 6	↖ 7	← 7
C	0	↑ 1	↖ 2	↑ 3	↖ 4	↑ 5	↑ 6	↑ 6	↑ 6	↖ 7	← 7	← 7

A2M SUPREME

EDIT DISTANCE

		G	U	M	B	O
	0	1	2	3	4	5
G	1	0	1	2	3	4
A	2	1	1	2	3	4
M	3	2	2	1	2	3
B	4	3	3	2	1	2
O	5	4	4	3	2	1
L	6	5	5	4	3	2

MATH
MYTH
MET
PET
MAT

G U M B O
G A M B O L

CHANGE DELETE

$$m(x, y) = \min \begin{cases} m(x-1, y-1) + 0 \; ; \text{ if matched} \\ m(x-1, y) + 1 \; ; \text{ otherwise} \\ m(x, y-1) + 1 \; ; \text{ otherwise} \end{cases}$$

ABCDBA

$\boxed{\text{ABCBA}}$ =

		S	A	T	U	R	D	A	Y
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
U	2	1	21	32	42	3	4	5	6
N	3	2	2	2	3	43	45	56	67
D	4	3	3	3	34	54	3	4	5
A	5	4	3	4	4	4	4	3	4
Y	6	5	4	4	5	5	5	4	3

S A T U R D A Y

S U N D A Y

INSERT CHANGE

DYNAMIC KNAPSCAK PROBLEM:

MAX WEIGHT: 10

ITEM: 4

i	1	2	3	4
Value i	10	40	30	50
weight i	5	4	6	3

$$V[i,w] = \max \begin{cases} V[i-1,w] \\ V[i] + V[i-1, w-w_i] \end{cases}$$

V[i,w]	0	1	2	3	4	5	6	7	8	9	10
i = 0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10	10	10	10
2	0	0	0	0	40	40	40	40	40	50	50
3	0	0	0	0	40	40	40	40	40	50	70
4	0	0	0	50	50	50	50	90	90	90	90

i	1	2	3	4	5
value i	25	20	15	40	50
weight i	3	2	1	4	5

max weight = 6

V[i,w]	0	1	2	3	4	5	6
i = 0	0	0	0	0	0	0	0
1	0	0	0	25	25	25	25
2	0	0	20	25	25	45	45
3	0	15	20	35	35	40	60
4	0	15	20	35	40	55	60
5	0	15	20	35	40	55	70

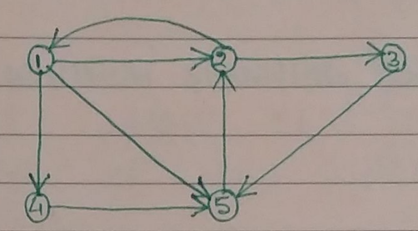
A2M SUPREME

GRAPH THEORY

VERTEX: a set of points

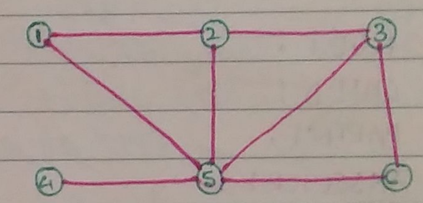
EDGE: connection between vertics.

DIRECTED GRAPH:

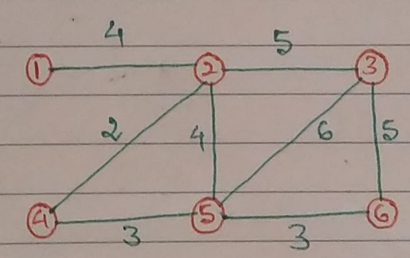


GRAPH : 1

UNDIRECTED GRAPH:



WEIGHTED GRAPH:



DEGREE: # of edge ends at that vertex.

DIRECTED GRAPH:

1) INDEGREE: # OF INCOMMING EDGES

11) OUT DEGREE: # OF OUTGOING EDGES

INDEGREE = OUTDEGREE

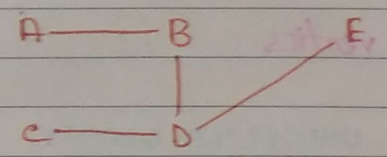
DEGREE

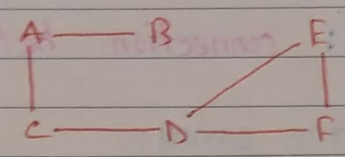
1 - 1
2 - 4
3 - 3
4 - 2
5 - 4
6 - 2

DEGREE / GRAPH 1

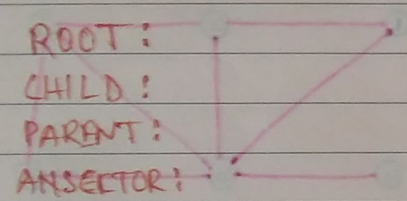
	IN	OUT
1	1	3
2	2	2
3	1	1
4	1	1
5	3	1
	8	8

TREE: A GRAPH WITH NO SIMPLE CIRCUIT.





ROOT:
CHILD:
PARENT:
ANSECTOR:



SPANNING TREE: A spanning tree OF A CONNECTED GRAPH G ~~CAN~~ ~~ADJO~~ IS A MAXIMAL SET OF EDGES OF G, THAT CONTAIN NO CYCLE. OR MINMUM SET OF EDGES THAT CONNECT ALL VERTICS.

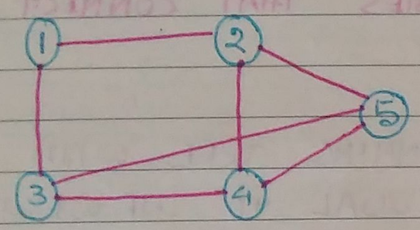
MINIMUM SPANNING TREE: A SPANNING TREE WITH WEIGHT LESS THAN OR EQUAL TO THE WEIGHT OF EVERY OTHER SPANNING TREE.

No.

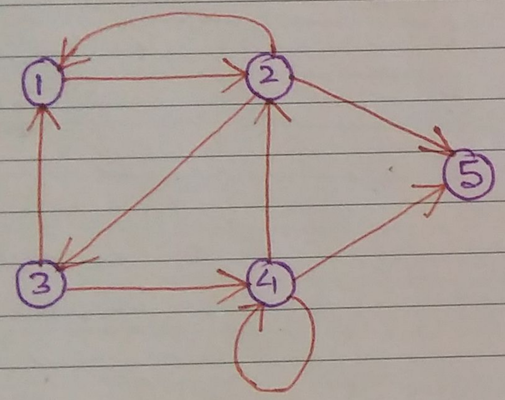
Date

GRAPH REPRESENTATION:

① ADJENCY LIST:



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



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

iii. EDGE LIST:

1. 1,2
2. 1,3
3. 2,4
4. 2,5
5. 3,4,
6. 3,5
7. 4,5

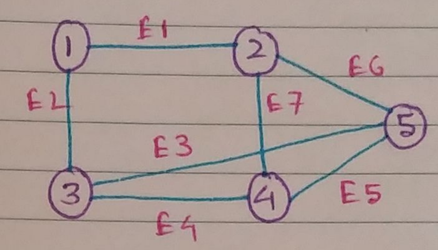
1. 1,2
2. 2,3
3. 2,5
4. 3,1
5. 3,4
6. 4,2
7. 4,4
8. 4,5

A2M SUPREME

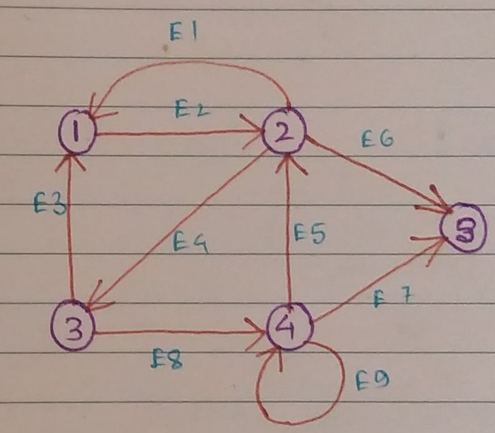
No.

Date

II. ADJENCY MATRIX:



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



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

IV) AINCIDENCE MATRIX:

	E1	E2	E3	E4	E5	E6	E7
1	1	1	0	0	0	0	0
2	1	0	0	0	0	1	1
3	0	1	1	1	0	0	0
4	0	0	0	1	1	0	1
5	0	0	1	0	1	1	0

	E1	E2	E3	E4	E5	E6	E7	E8	E9
1	-1	1	-1	0	0	0	0	0	0
2	1	-1	0	1	-1	1	0	0	0
3	0	0	1	-1	0	0	0	1	0
4	0	0	0	0	1	0	1	-1	2
5	0	0	0	0	0	-1	-1	0	0

A2M SUPREME

DATA STRUCTURE:

QUEUE : FIRST IN FIRST OUT

```
void Enque (int item)
{
    queue [front] = item;
    front ++;
}

int Deque ()
{
    int item = queue [rear];
    rear ++;
    return item;
}

int front =1, rear = 1;

int QUEUE [ ];
```

FRONT: 1 REAR: 1

	1	2	3	4	5	6	7	8	9	10
QUEUE:	~~1~~	~~4~~	~~5~~	6	5					

Enque (1)	F:2, R=1
Enque (4)	F:3, R=1
Enque (5)	F:4, R=1
Deque () = 1	F:4, R=2
Deque () = 4	F:4, R=3
Enque (6)	F:5, R=3
Deque () = 5	F:5, R=4
Enque (5)	F=6, R=4

STACK : FIRST IN LAST OUT

```
int top = 0; int stack [10];

void pust (int item)
{
    stack [top] = item;
    top++;
}

int pop ()
{
    int item = stack[--top];
    return item;
}
```

TOP: 0

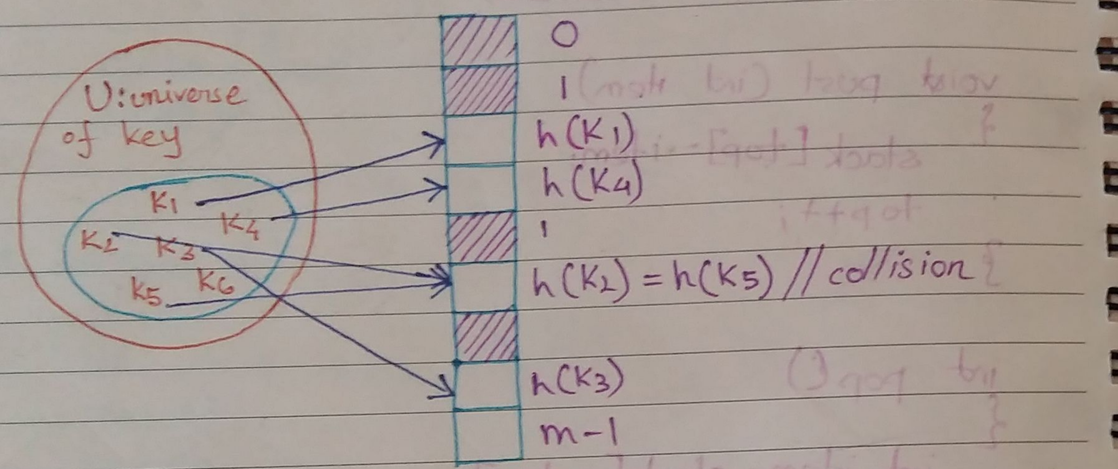
STACK:

	1	2	3	4						
	1	~~4~~	~~8~~	~~4~~						

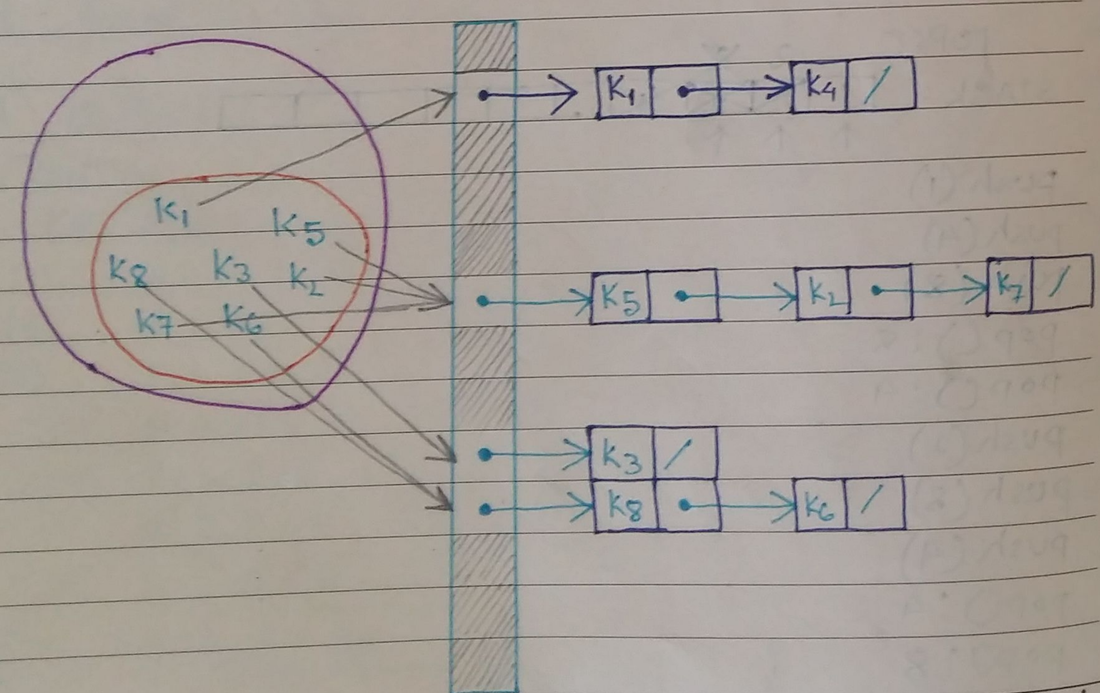
push (1)
push (4)
push (8)
pop () : 8
pop () : 4
push (2)
push (8)
push (4)
pop() : 4
pop() : 8

HASH TABLE:

HASH FUNCTION: h to compute the slot from the key.

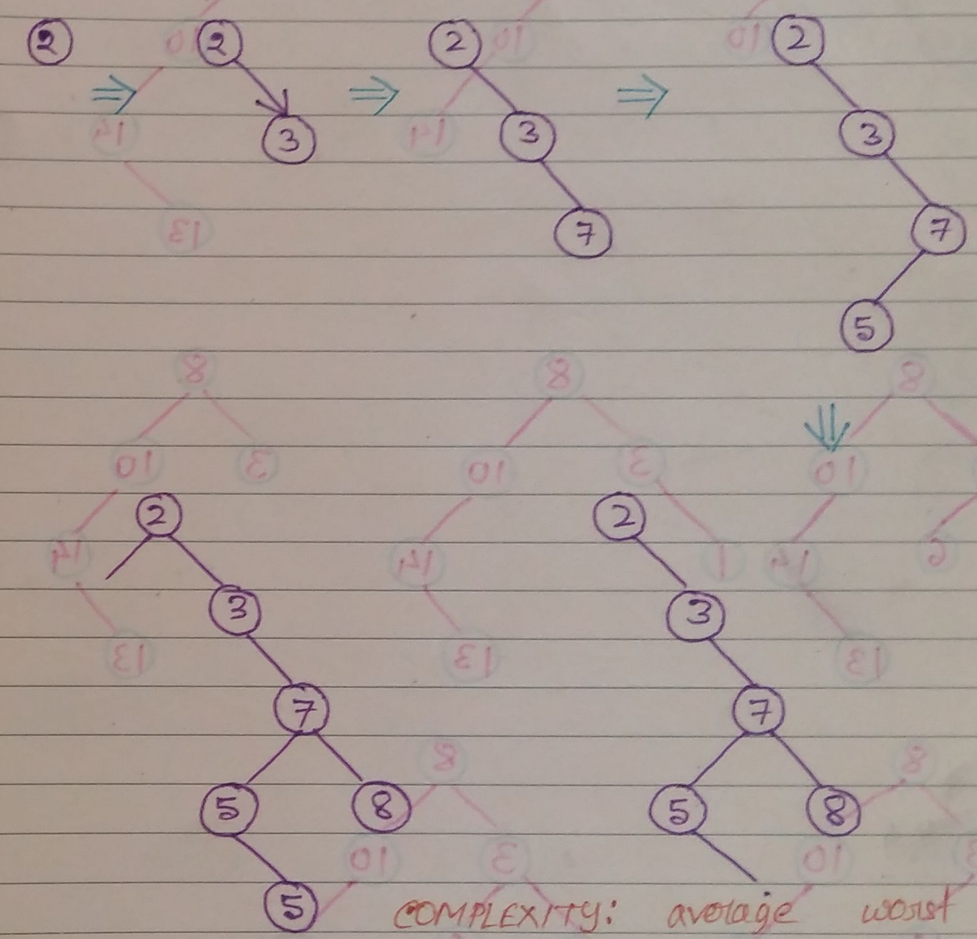


COLLISION RESOLUTION BY CHAINING:



BINARY SEARCH TREE

2, 3, 7, 5, 5, 8

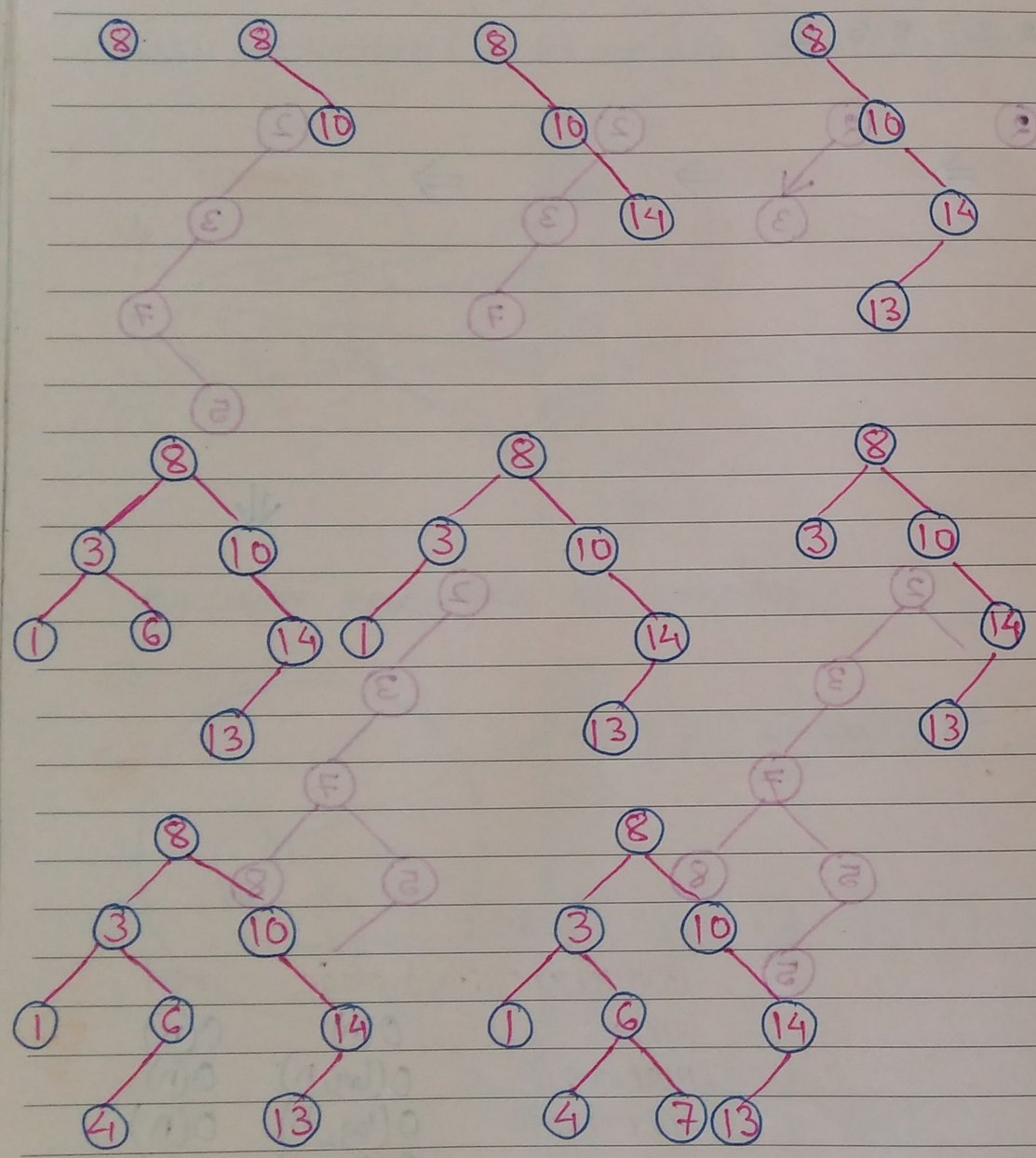


COMPLEXITY:	average	worst
SPACE:	$O(n)$	$O(n)$
INSERT:	$O(\log_2 n)$	$O(n)$
DELETE:	$O(\log_2 n)$	$O(n)$
SEARCH:	$O(\log_2 n)$	$O(n)$

No.

Date

8 10 14 13 3 1 6 4 7

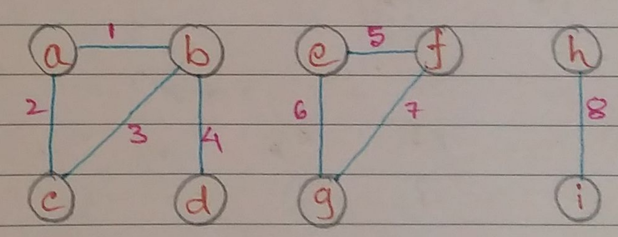


No.

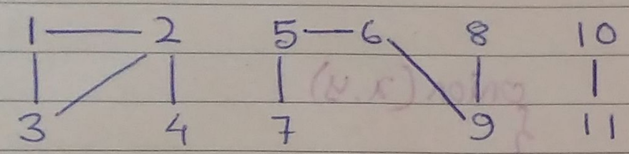
Date

DISJOINT SET:

```
make_set(x)
{
    P(X) = X
    rank(x) = 0
}

Find_set (X)
{
    if X ≠ P(X)
        P(X) = Find_set(P(X));
    return P(X);
}

union(x,y)
{
    Link(Find_set(x), Find_set(y))
}

Link (X, Y)
{
    if rank (x) > rank(y)
        P(y) = X
    else
        P(X) = y
    if rank (x) == rank(y)
        rank (y) ++;
}
```

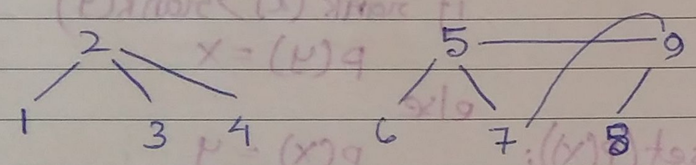


	a	b	c	d	e	f	g	h	i
Rank	0	0 1	0	0	0	0 1	0	0 1	0
P	a b	b h	c b	d b	e f	f	g f	h i	i h
(a,b)	b=0	b=1							
(a,c)	b=0		b=0						
(b,d)		b=1		b=0					



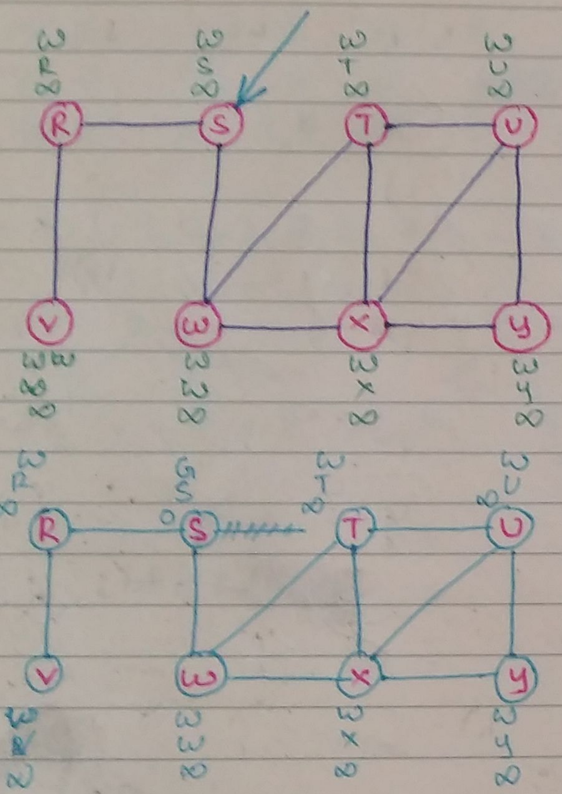
V	1	2	3	4	5	6	7	8	9	10	11
R	0	1	0	0	0	0	0	0	1	0	1
	12	2	23	24	5	65	75	89	96	11	11

1-2, 1-3, 2-3, 4-2, 7-5, 6-5, 8-9, 9-6, 10-11

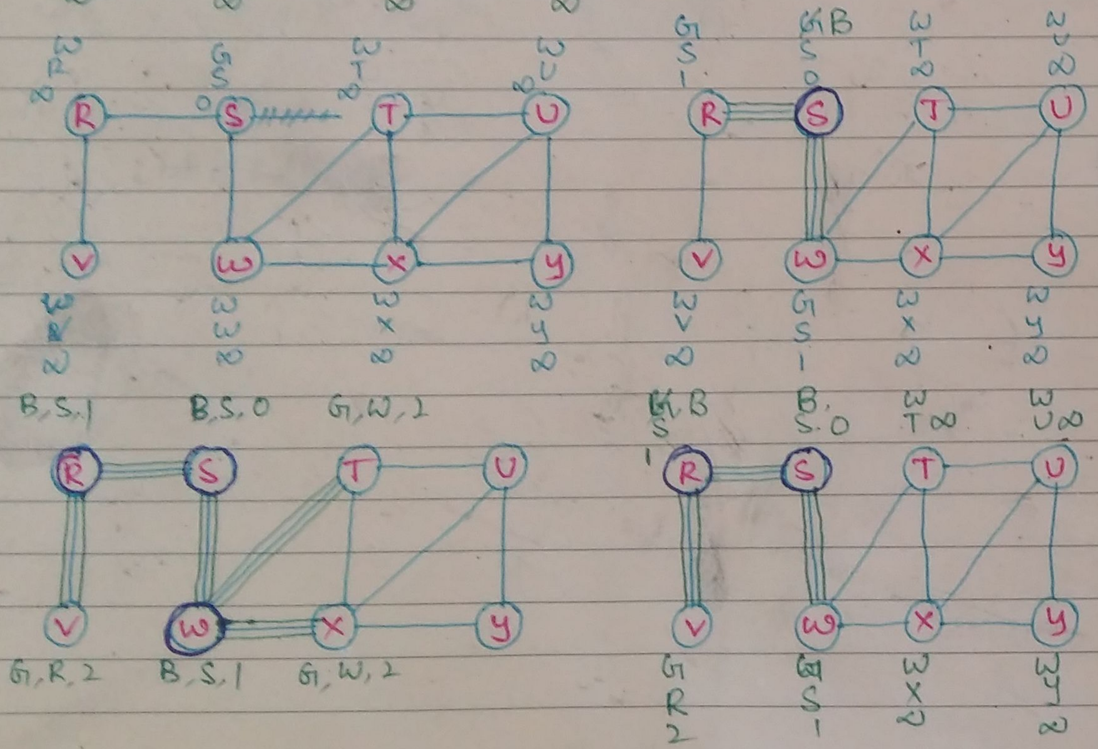


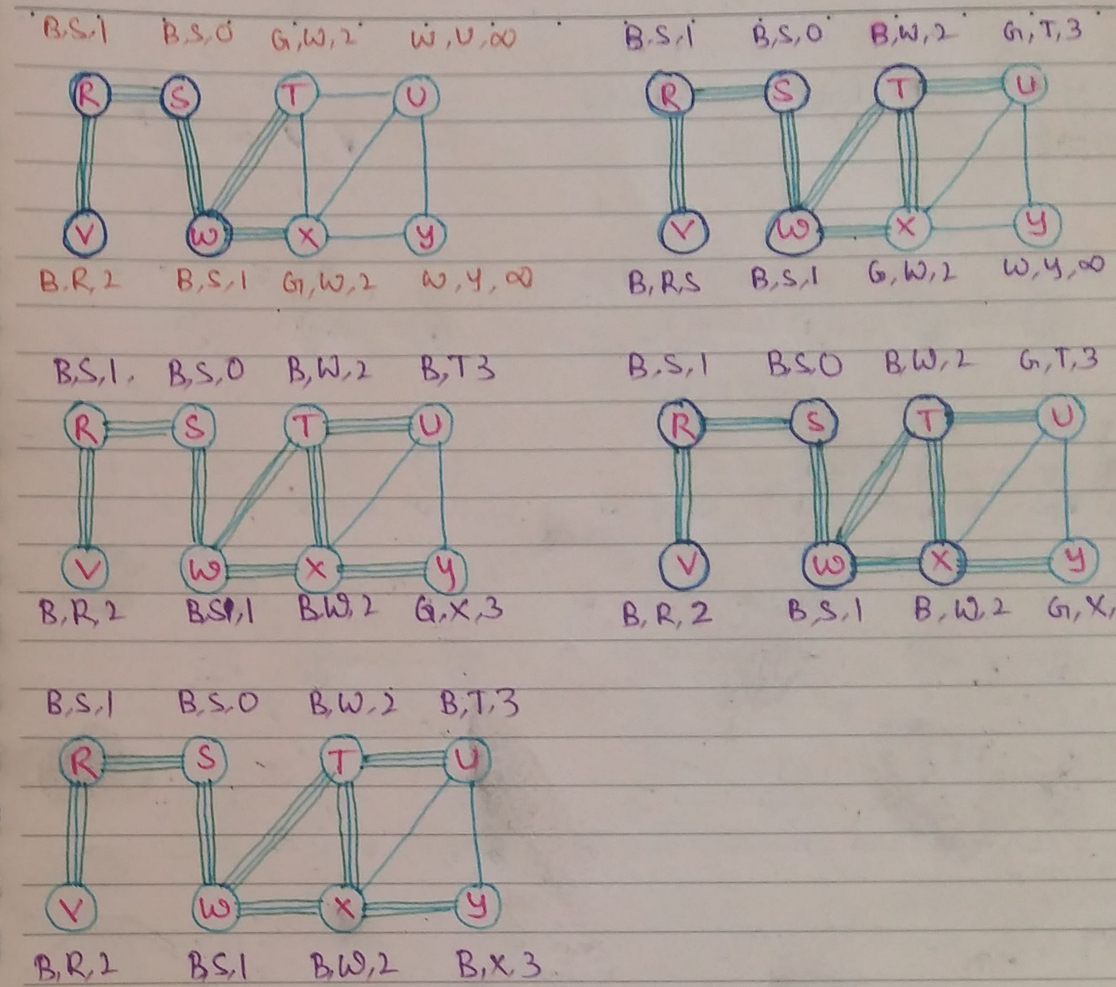
GRAPH SEARCH:

1. BREDTH FIRST SEARCH:



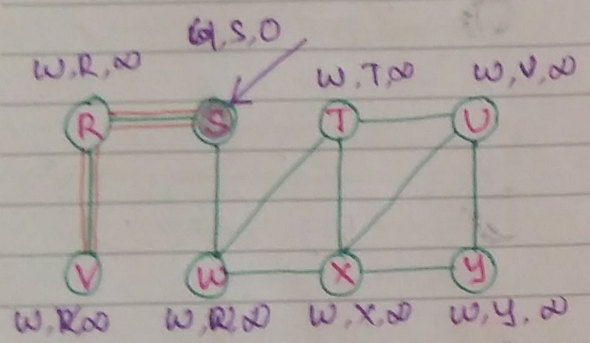
Q: ~~S, R, W, V, T, X, U, Y~~

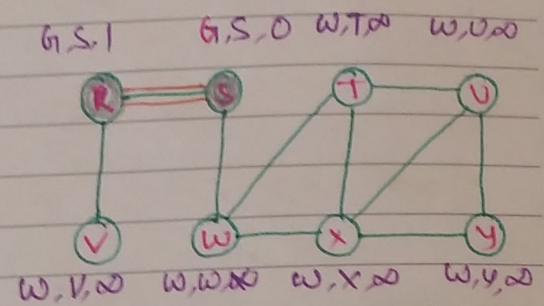


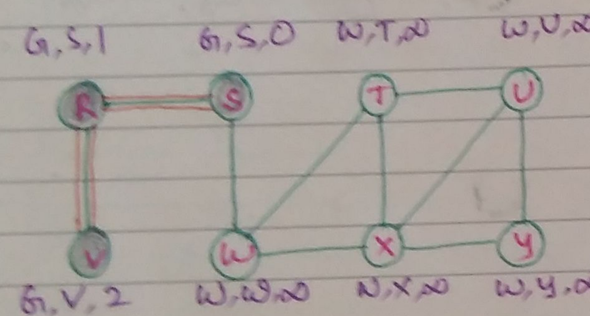


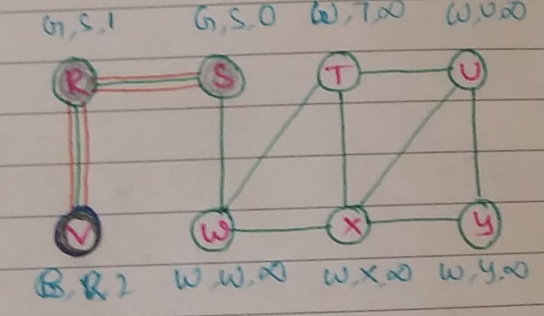
!! DEPTH FIRST SEARCH :

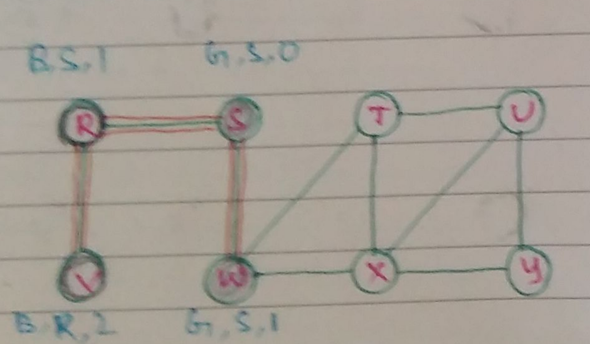
S: S, ~~R~~, ~~V~~, W, ~~T~~, ~~U~~, ~~Y~~, ~~X~~

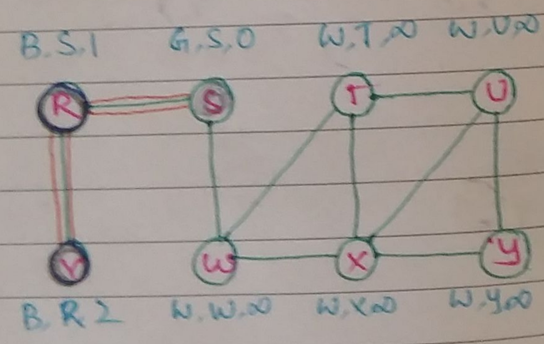


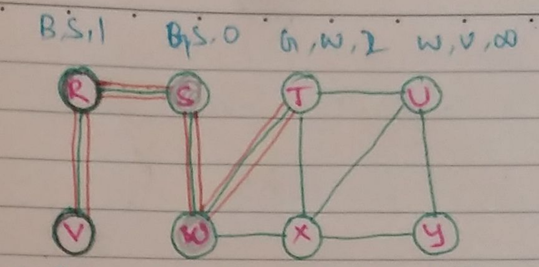


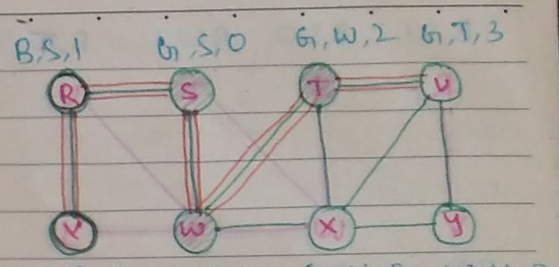


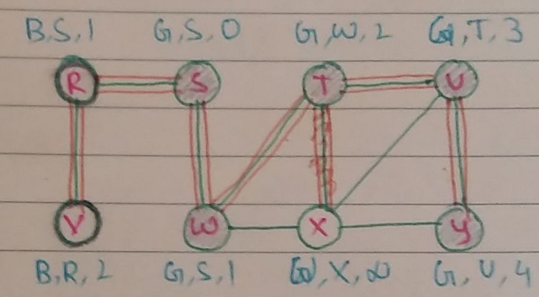


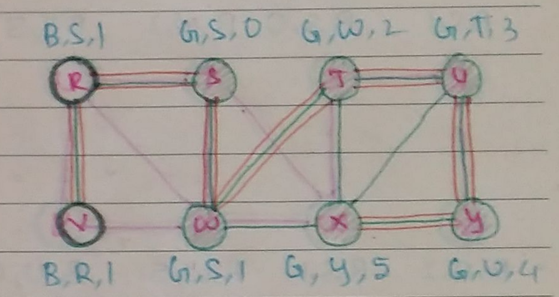


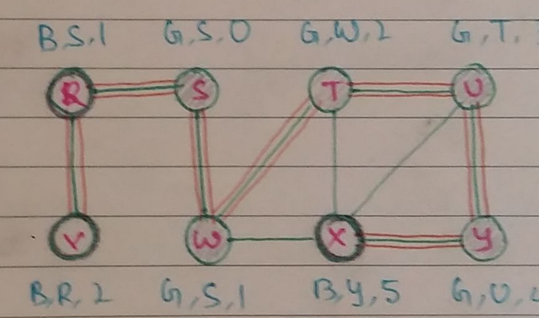


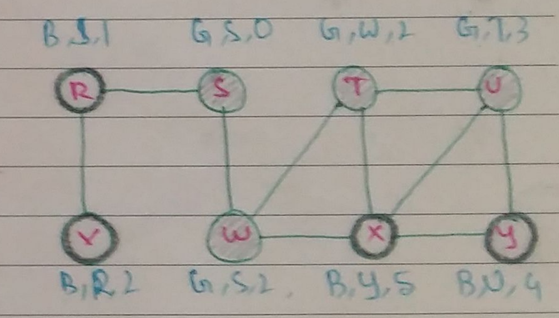


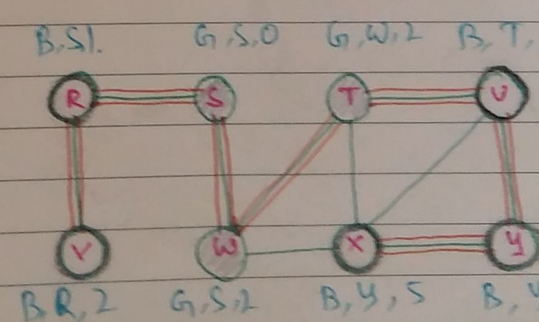


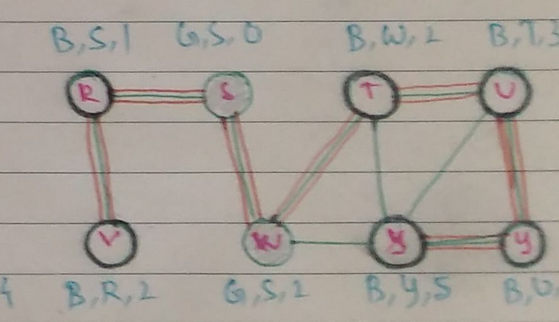




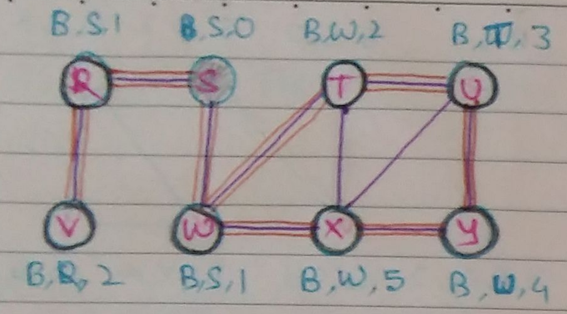


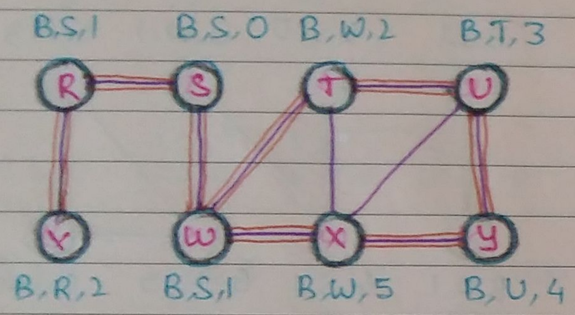






G


B,S,1
B,S,0
B,W,2
B,T,3
R
S
T
U
V
W
X
Y
B,R,2
B,S,1
B,W,5
B,U,4


B,S,1
B,S,0
B,W,2
B,T,3
R
S
T
U
V
W
X
Y
B,R,2
B,S,1
B,W,5
B,U,4

No.

Date

STRONGLY CONNECTED COMPONENTS:

INPUT: A DIRECTED GRAPH
OUTPUT: GRAPH WITH CONNECTED COMPONENTS:

A DIRECTED GRAPH IS CALLED STRONGLY CONNECTED COMPONENTS IF THERE IS A PATH IN EACH DIRECTION BETWEEN EACH PAIR OF VERTICS OF THE GRAPH.

CONNECTED COMPONENTS: A CONNECTED COMPONENTS OF AN UNDIRECTED GRAPH IS A SUB-GRAPH IN WHICH TWO VERTICS ARE CONNECTED TO EACH OTHER BY PATHS.

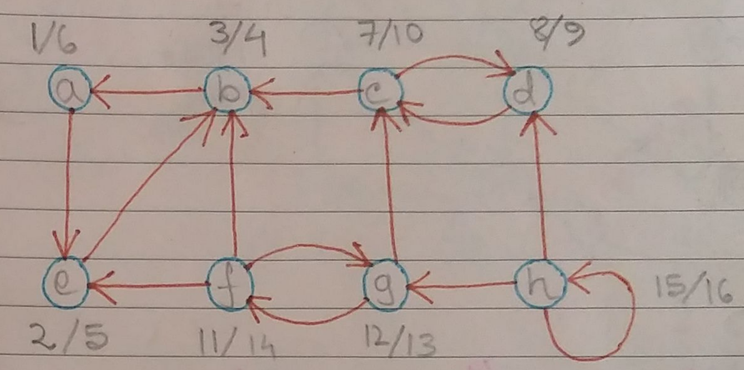
1. call DFS(G) to compute finishing times $f[u]$
2. compute G^T
3. call DFS (G^T), decreasing $f[u]$
4. output the vertices of each tree in the DFS forest formed in line 3

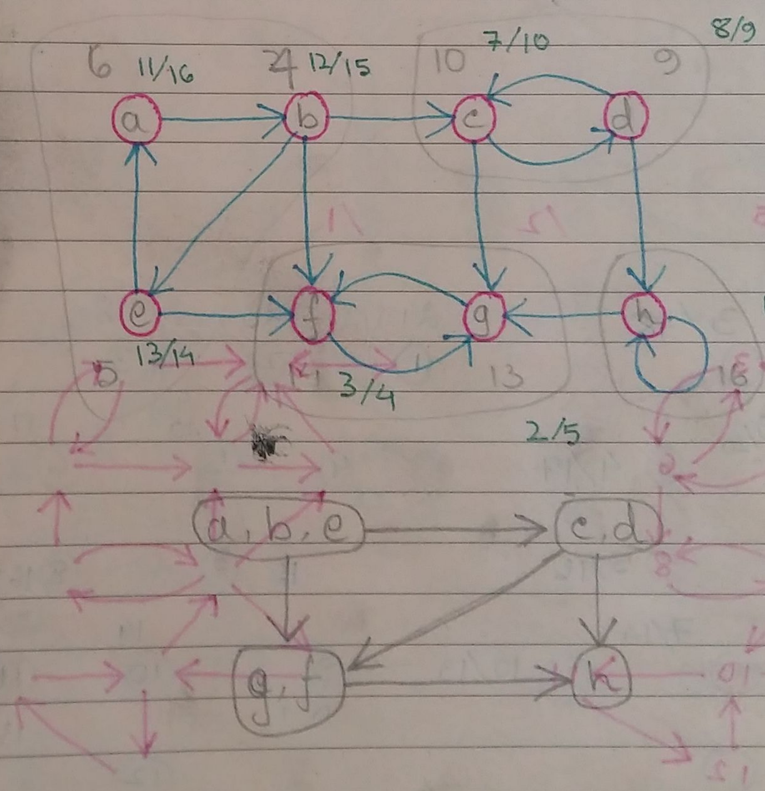
A2M SUPREME

No.

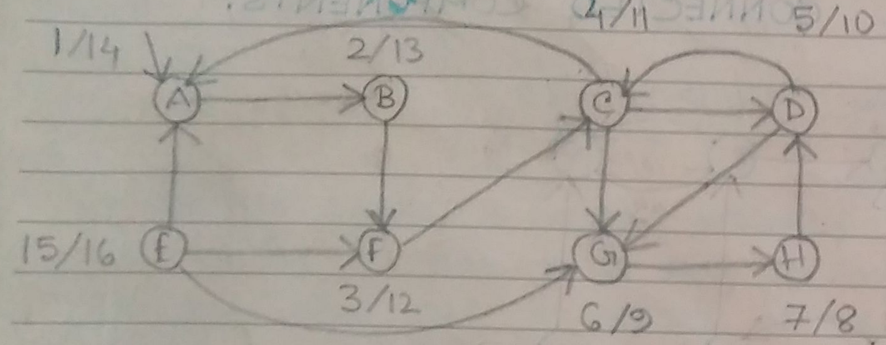
Date

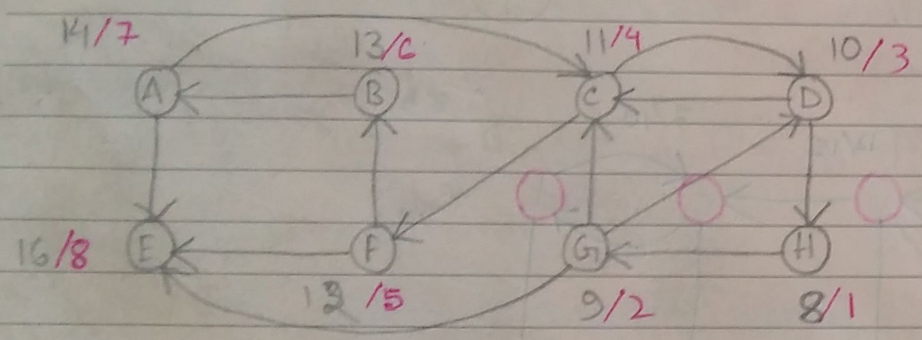
STRONGLY CONNECTED COMPONENTS:

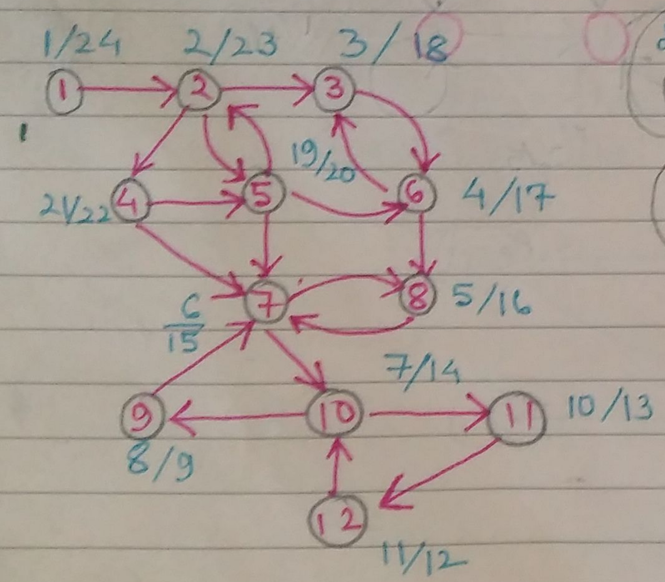


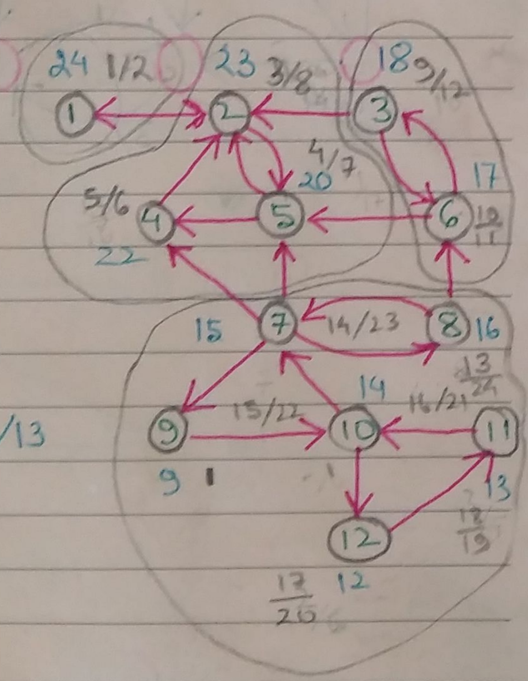


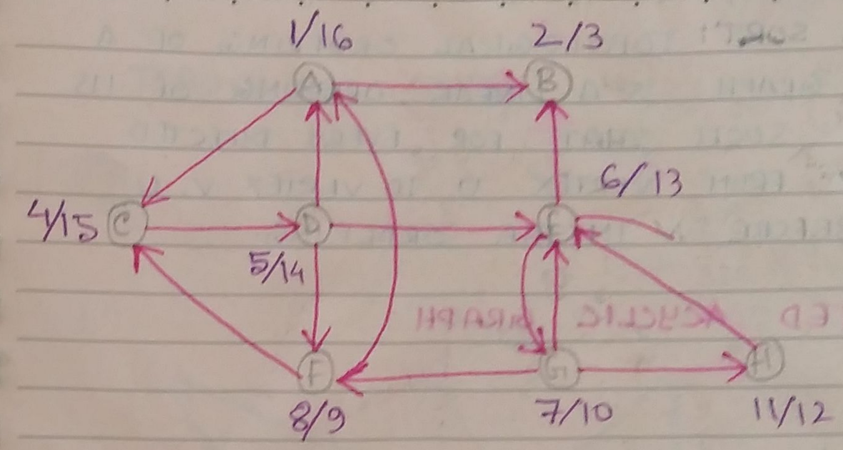
A2M SUPREME

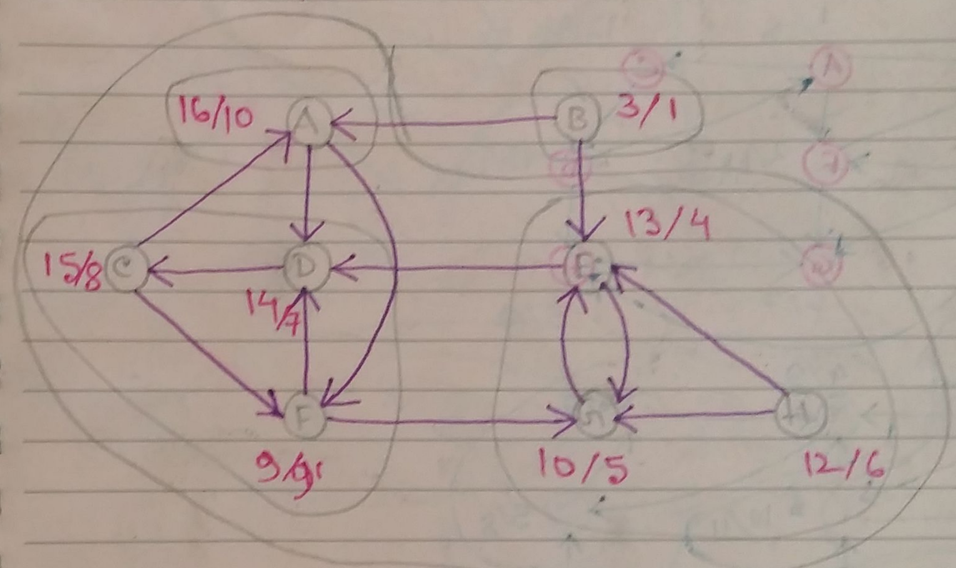

1/14
2/13
4/11
5/10
A
B
C
D
15/16
E
F
G
H
3/12
6/9
7/8

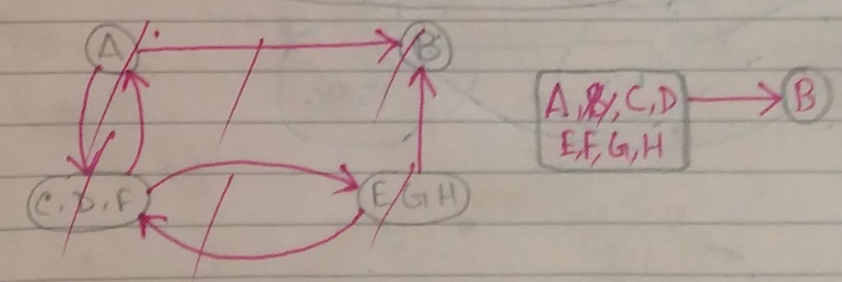

14/7
13/6
11/4
10/3
A
B
C
D
16/8
E
F
G
H
13/5
9/2
8/1


1/24
2/23
3/18
1
2
3
19/20
21/22
4
5
6
4/17
6/15
7
8
5/16
7/14
9
10
11
10/13
8/9
12
11/12


24 1/2
23 3/8
18 9/12
1
2
3
4/7
20
17
5/6
4
5
6
19/11
22
15
7
14/23
8
16
13/24
9
15/22
10
16/21
11
9
13
12
12/19
12
12/20

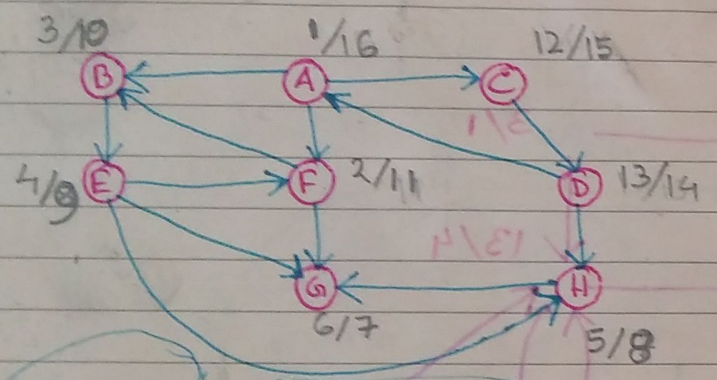

1/16
2/3
A
B
4/15
C
D
5/14
6/13
F
E
8/9
7/10
11/12

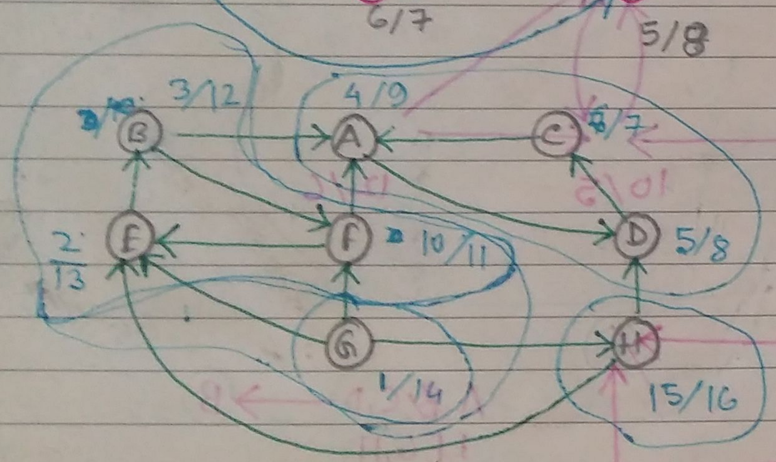

16/10
A
B
3/1
13/4
15/8
C
D
14/7
E
F
9/11
10/5
12/6


A
B
C, D, F
E, G, H
A, B, C, D
E, F, G, H
B

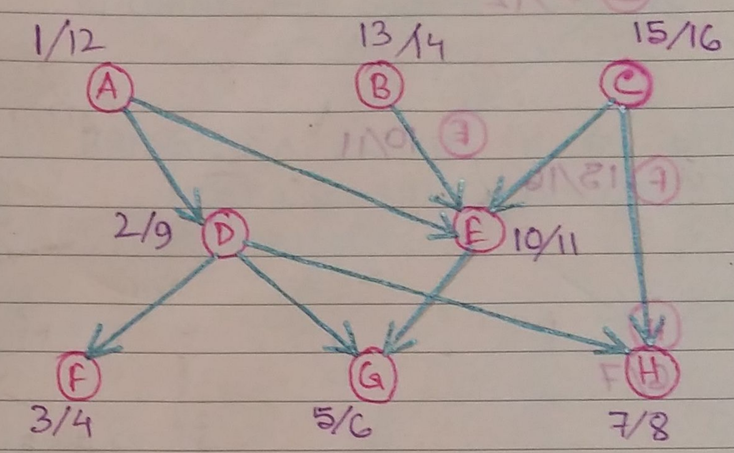
TOPOLOGICAL SORT: TOPOLOGICAL ORDERING OF A DIRECTED GRAPH IS A LINEAR ORDERING OF ITS VERTICES SUCH THAT FOR EVERY DIRECTED EDGE UV FROM VERTEX U TO VERTEX V, U COMES BEFORE V IN THE ORDERING.

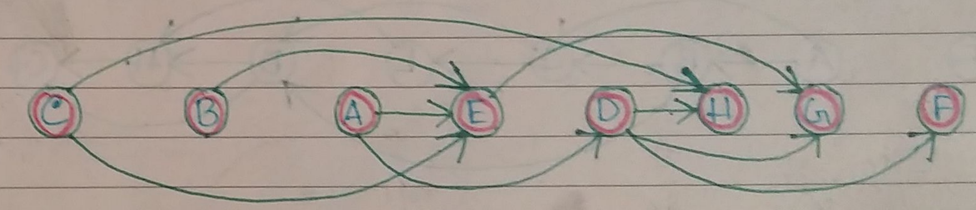
* DIRECTED ACYCLIC GRAPH

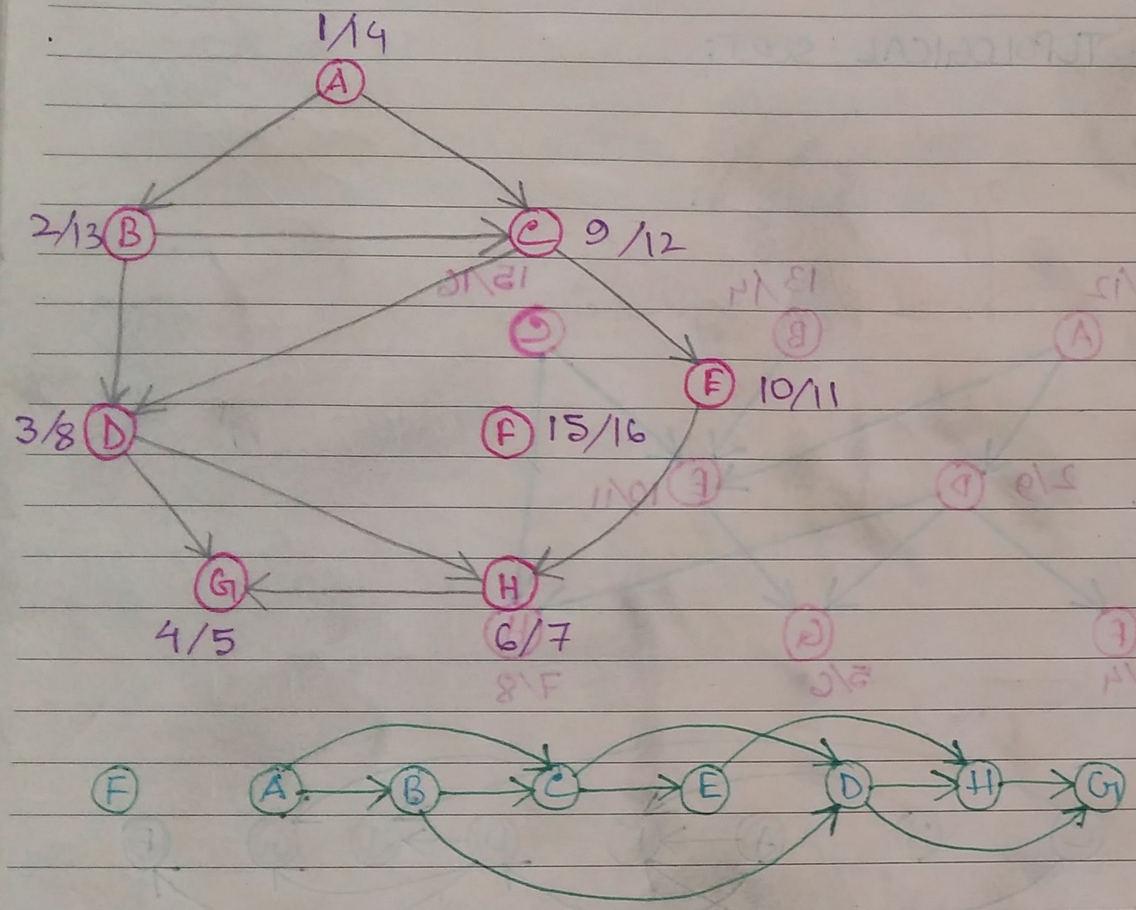




TOPOLOGICAL SORT:



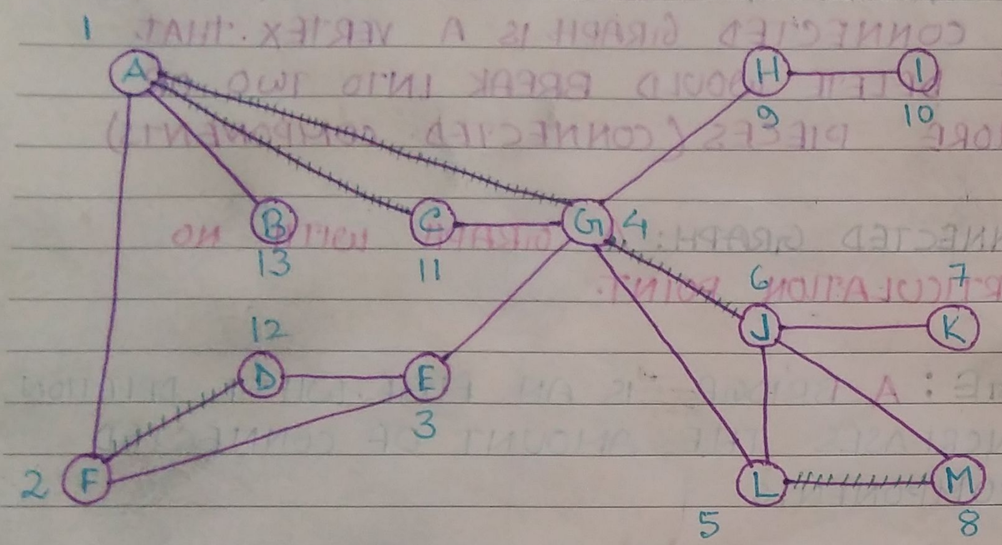




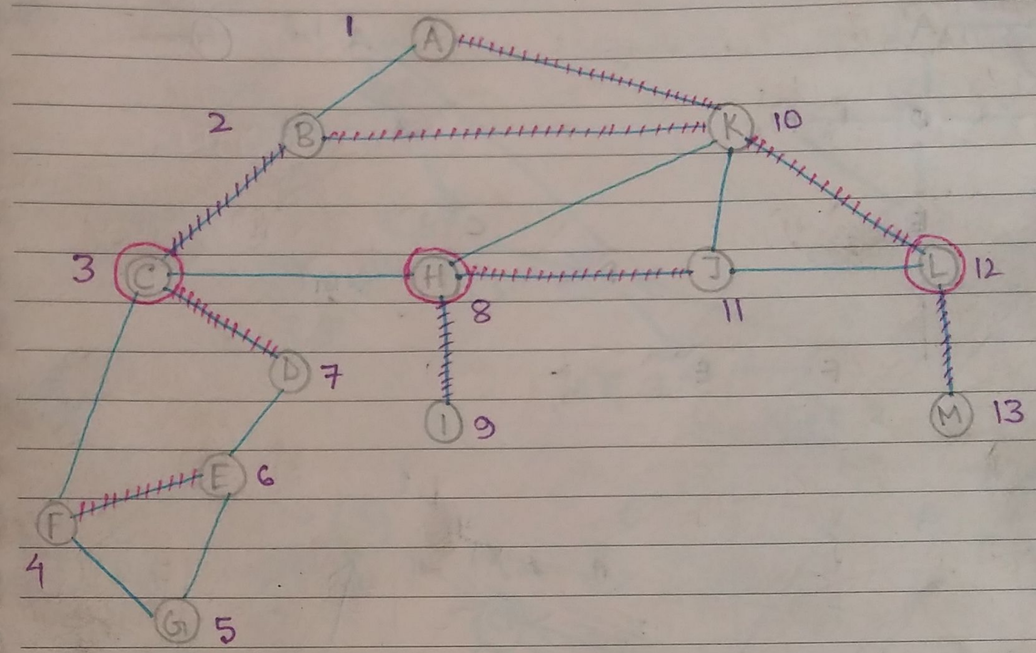
ARTICULATION POINT: AN ARTICULATION POINT IN A CONNECTED GRAPH IS A VERTEX THAT IF DELETE WOULD BREAK INTO TWO OR MORE PIECES (CONNECTED COMPONENTS)

BI-CONNECTED GRAPH: A GRAPH WITH NO ARTICULATION POINT.

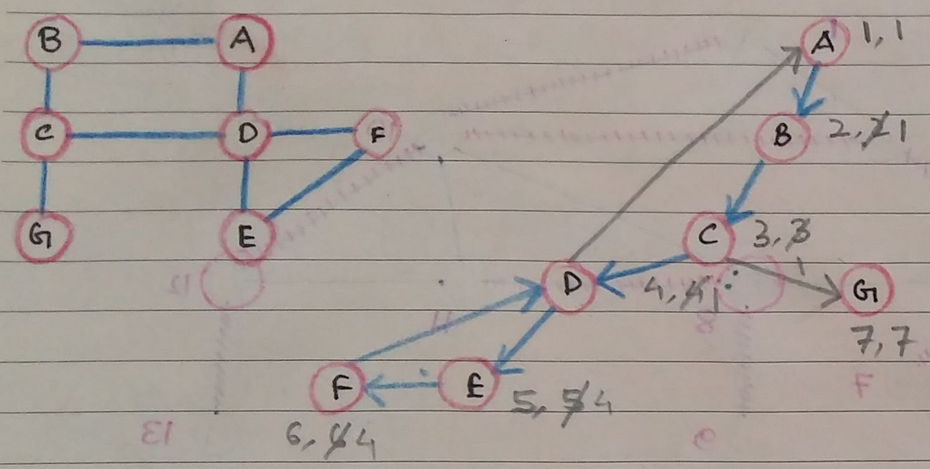
BRIDGE: A BRIDGE IS AN EDGE WHOSE DELETION INCREASES THE AMOUNT OF CONNECTED COMPONENTS.

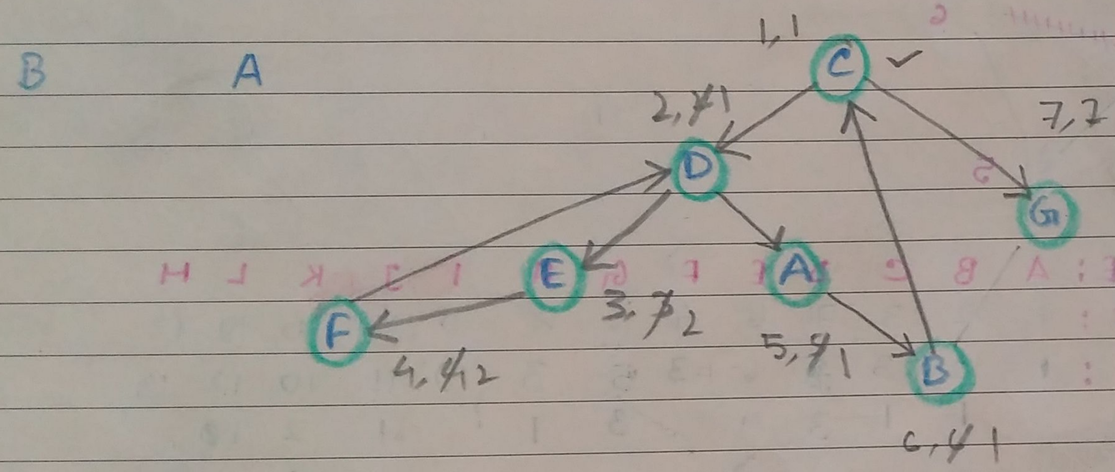


VERTEX	A	B	C	D	E	F	G	H	I	J	K	L	M
DNF	1	13	11	12	3	2	4	9	10	6	7	5	8
L	1	13	11	12	3	2	4	9	10	6	7	5	5
			1	2	1	1	1			4		4	5
LOW	1	13	1	2	~~3~~	1	1	9	10	4	7	5	8

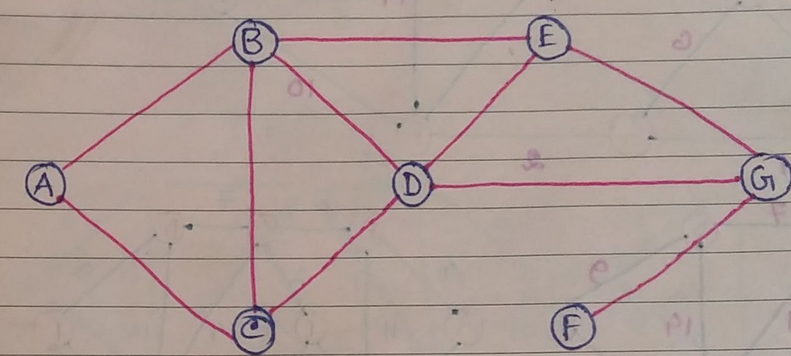


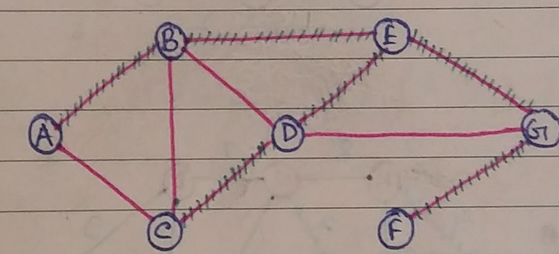
NODE :	A	B	C	D	E	F	G	H	I	J	K	L	M
DNF :	1	2	3	7	6	4	5	8	9	11	10	12	13
LOW :	1	2	3	~~1~~	6	3	5	3	9	11	10	12	13
		1	1	3	4		3	1		1	2	10	
					3						1		

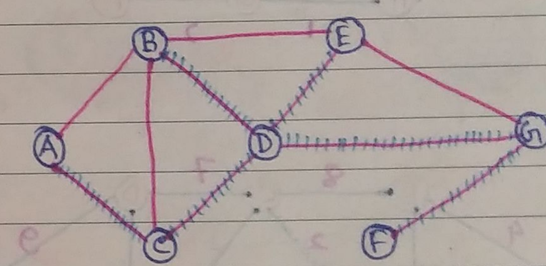


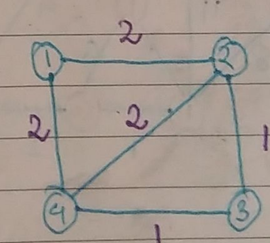


SPANNING TREE:







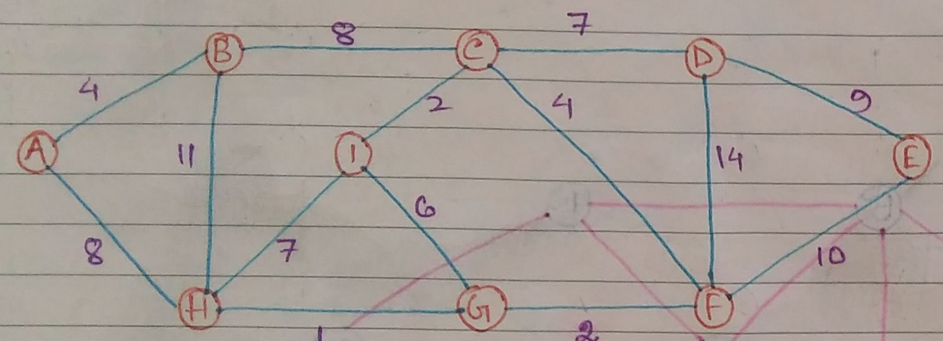


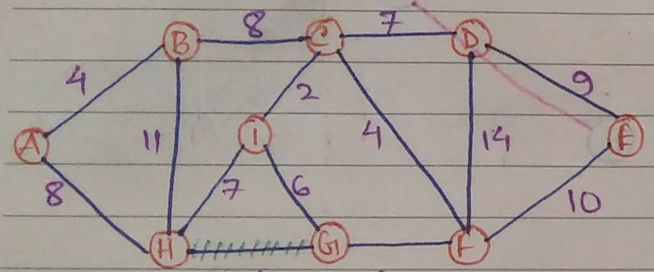
No.

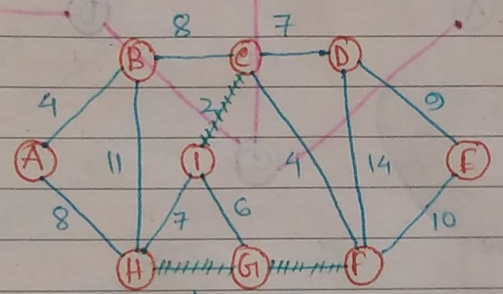
Date

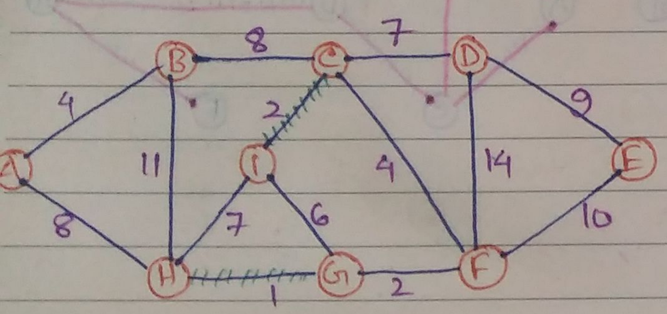
MINIMUM SPANNING TREE:

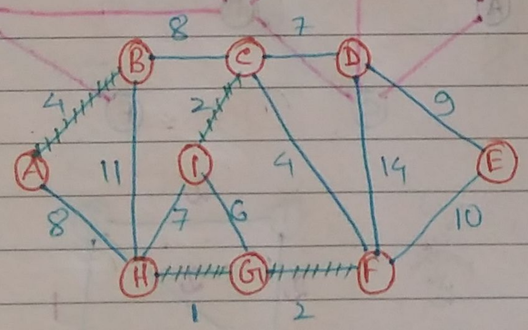
KRUSKEL'S ALGORITHM:





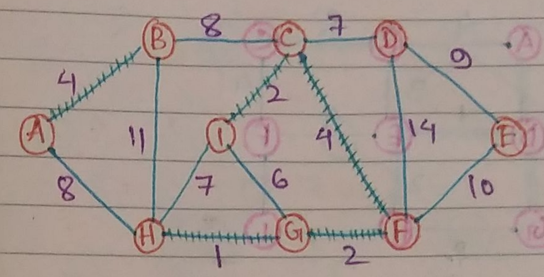


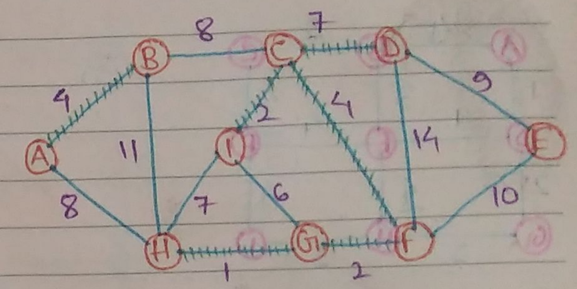


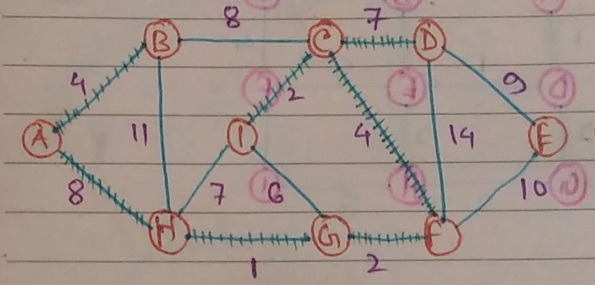


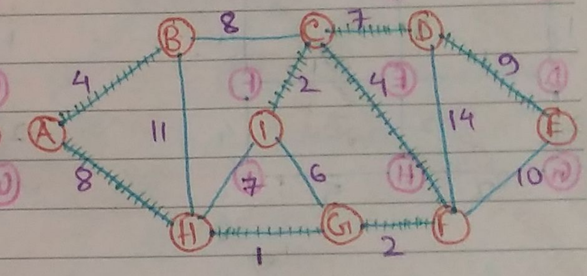
No.

Date

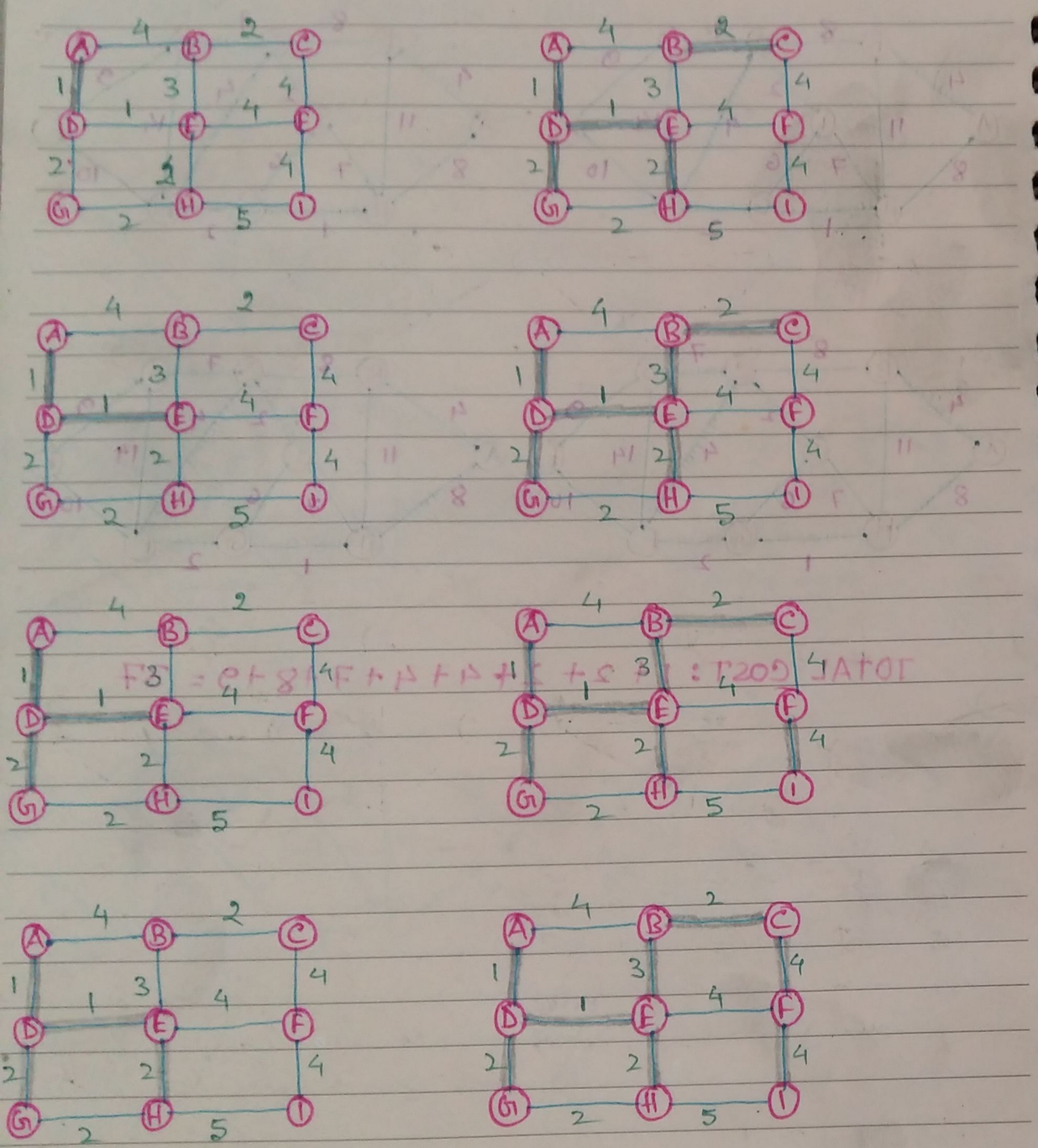








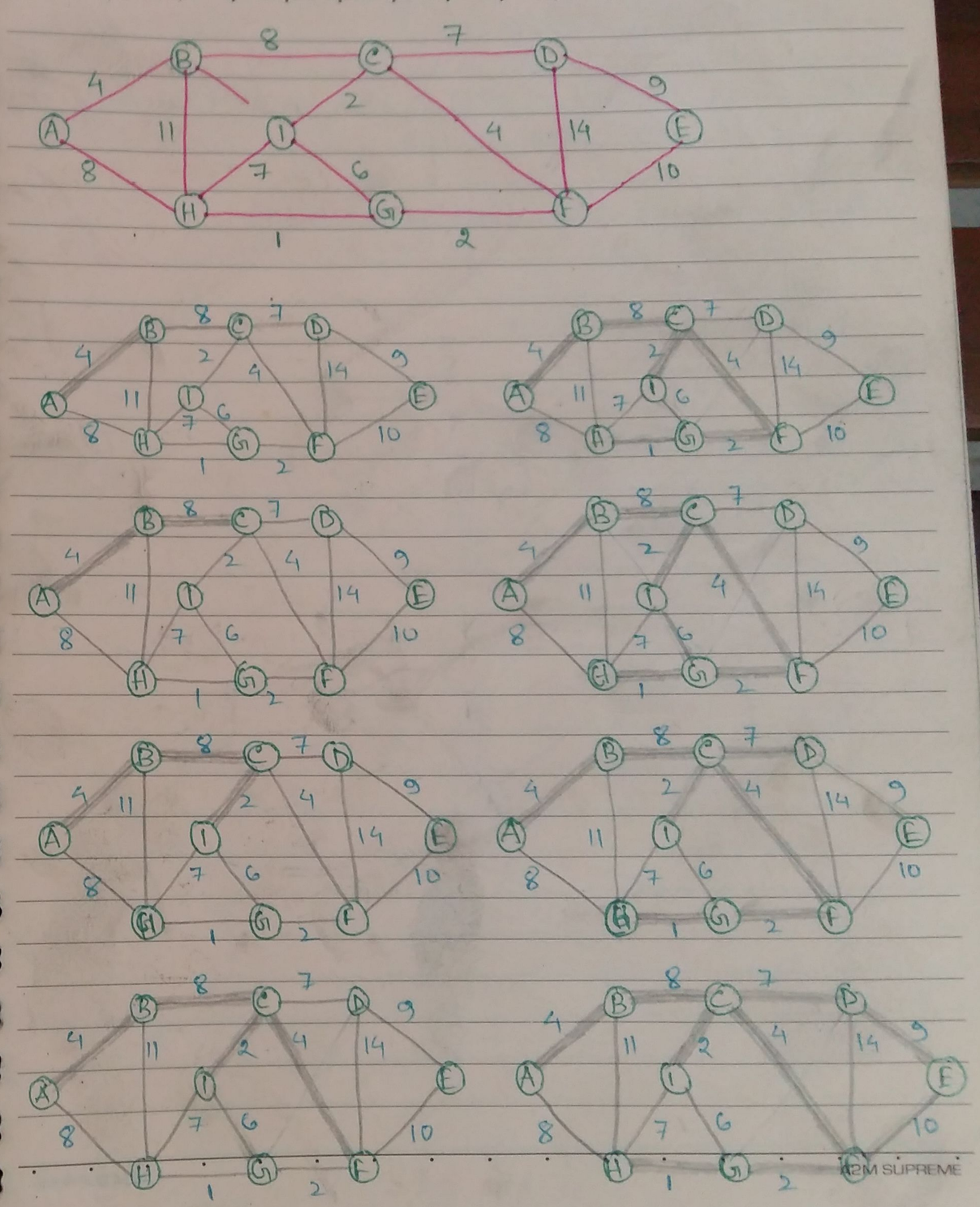
TOTAL COST: 1 + 2 + 2 + 4 + 4 + 7 + 8 + 9 = 37



TOTAL COST : 1+1+2+2+2+3+4+4 = 19

A2M SUPREME

PRIMS ALGORITHM



A2M SUPREME

No.

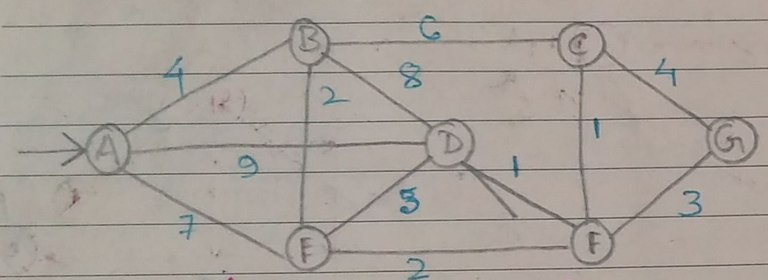
Date

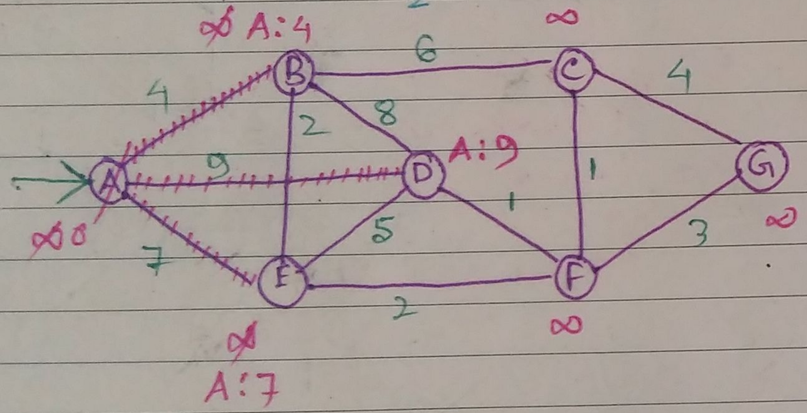
SHORTEST PATH PROBLEM:

INPUT: GIVEN A GRAPH, A START & END VERTEX. FIND SHOREST PATH FROM START TO END.

OUTPUT: SHORTES PATH FROM START TO END.

DIJKSTRA'S ALGORITHM:

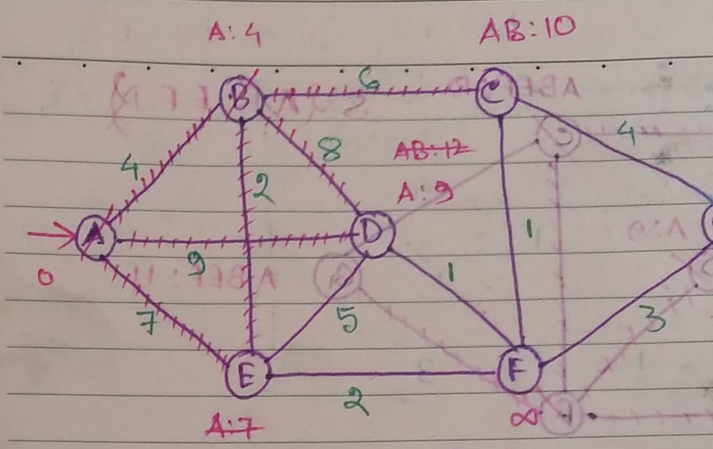




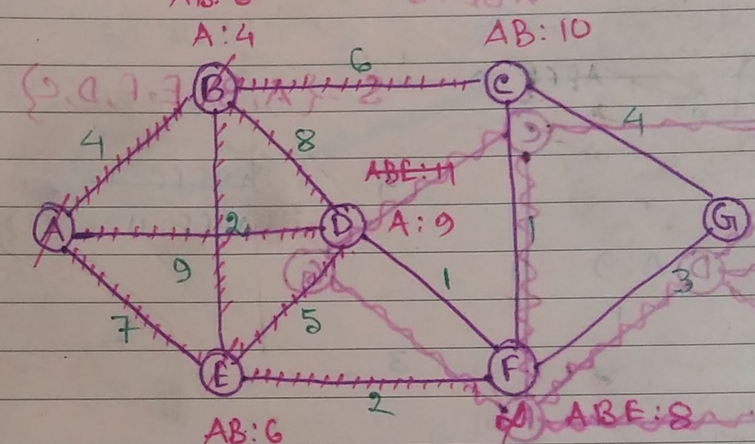
S={A, B}

No.

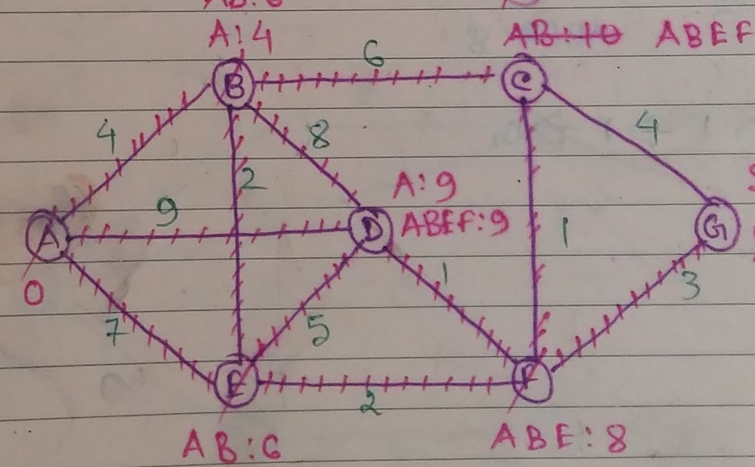
Date



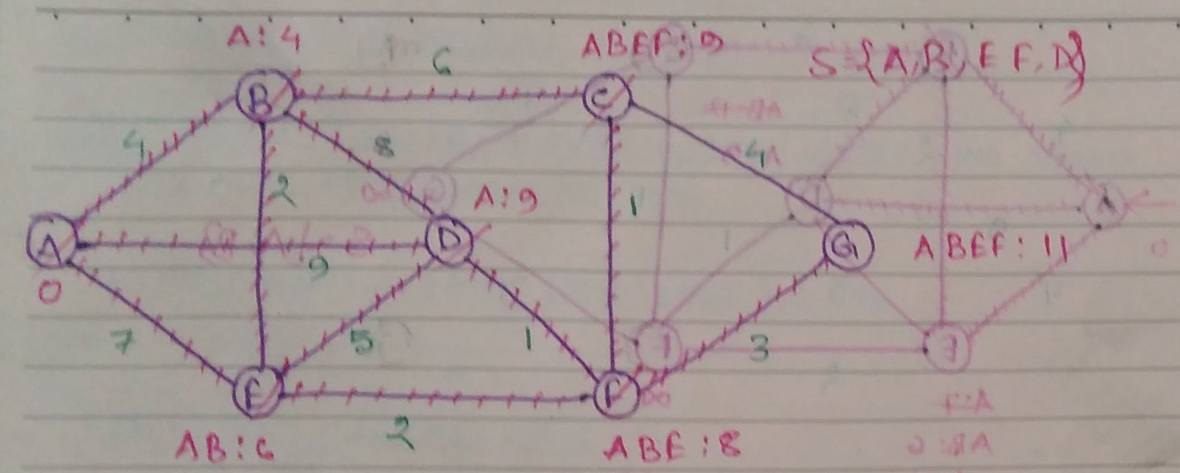
S={A, B}

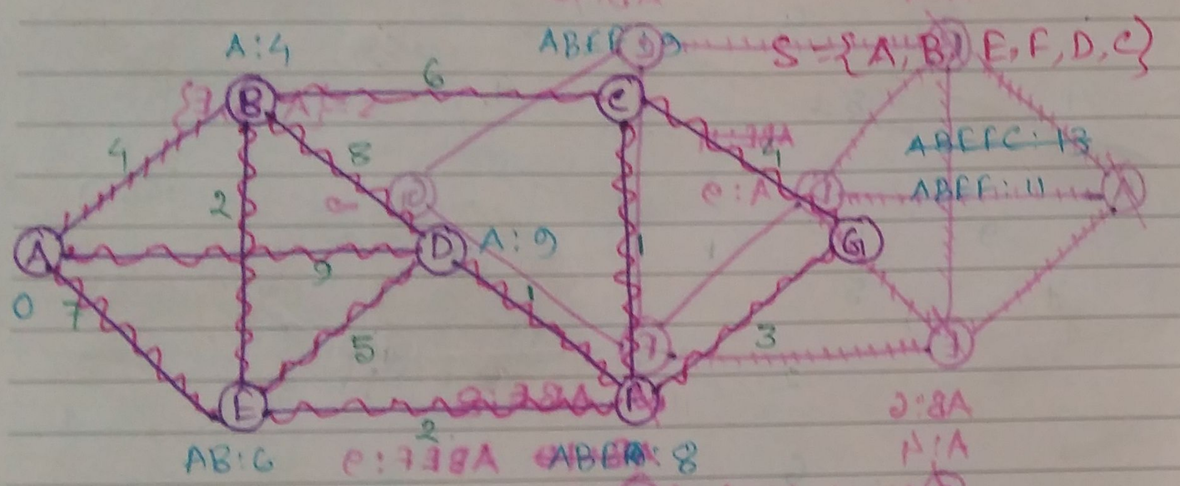


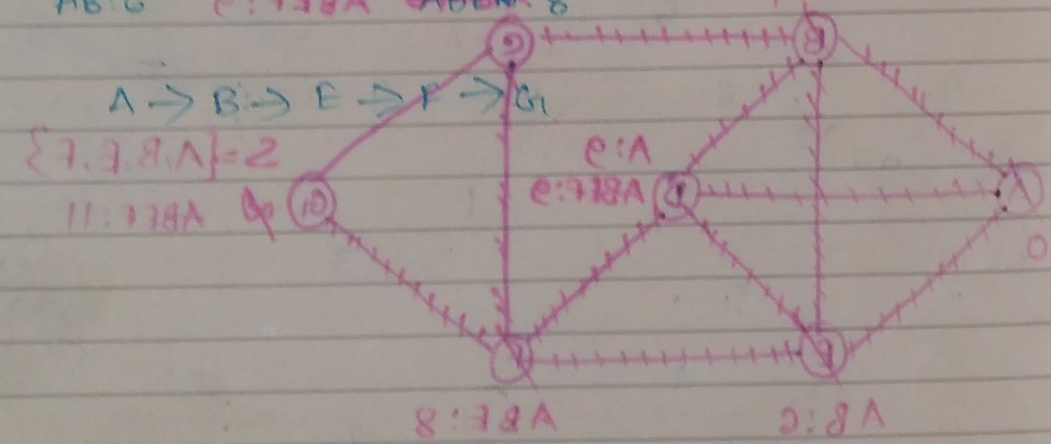
S={A, B, E}

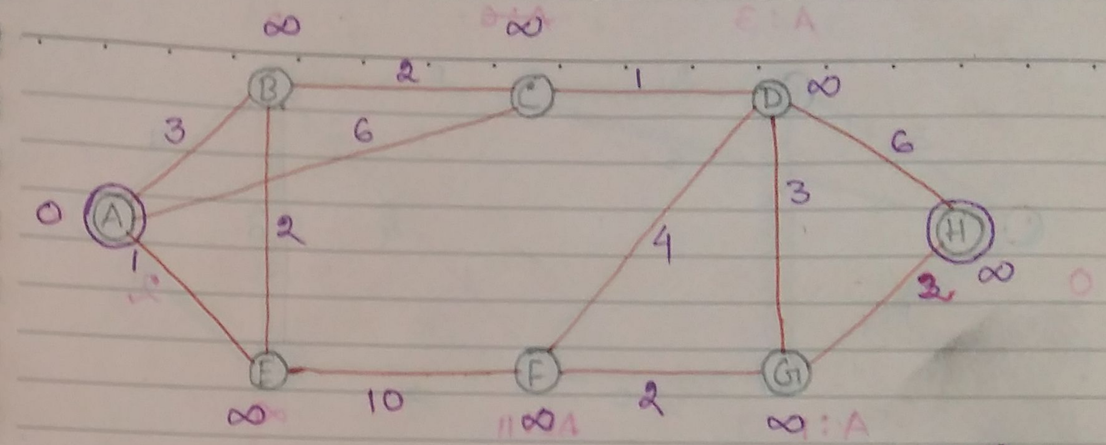


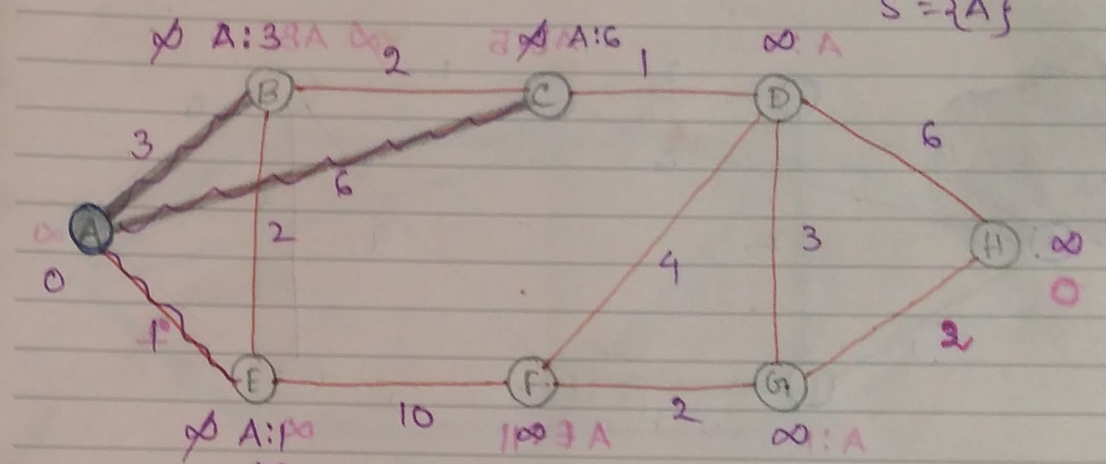
S={A, B, E, F}

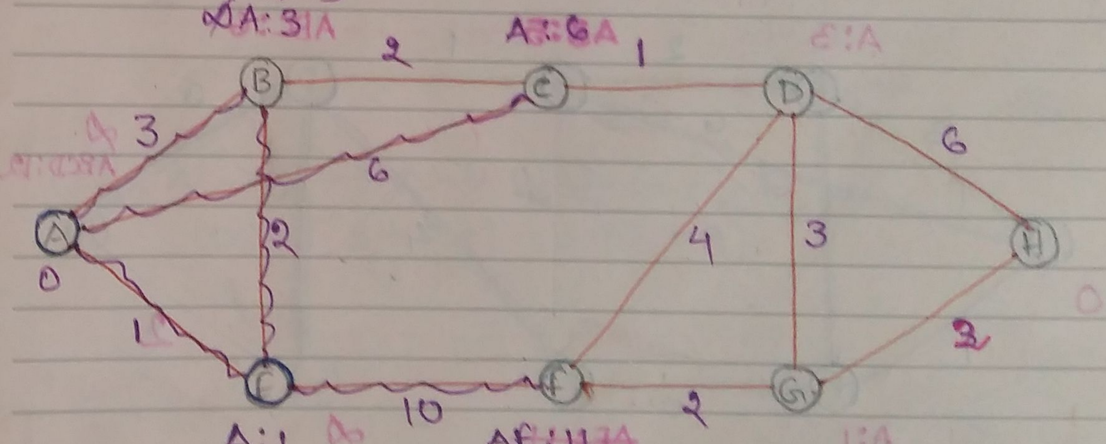

A:4
ABEF:9
S={A,B,E,F,D}
A:9
ABEF:11
AB:6
ABE:8

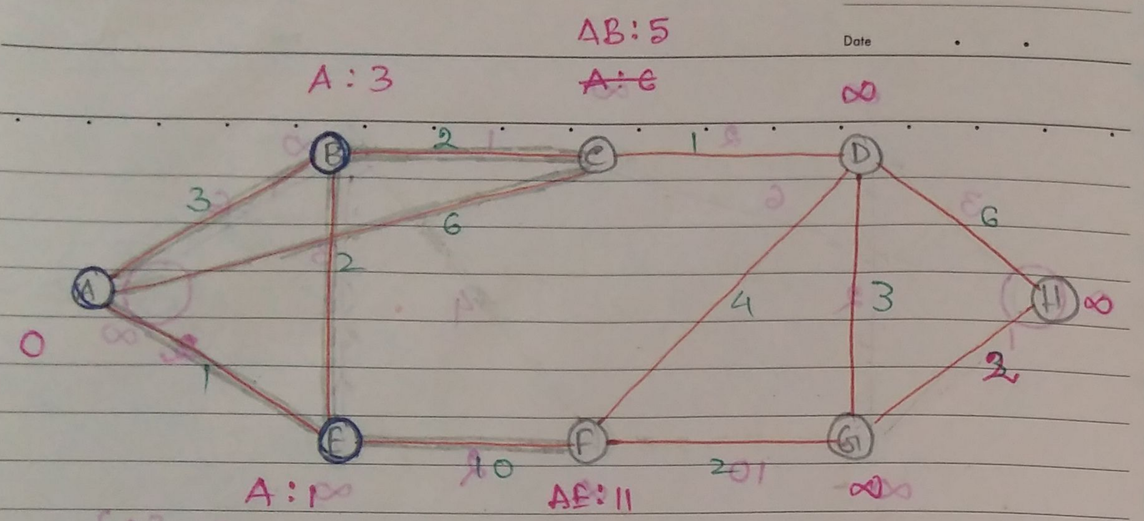

A:4
ABEF:9
S={A,B,E,F,D,C}
ABEFC:13
ABEF:11
A:9
AB:6

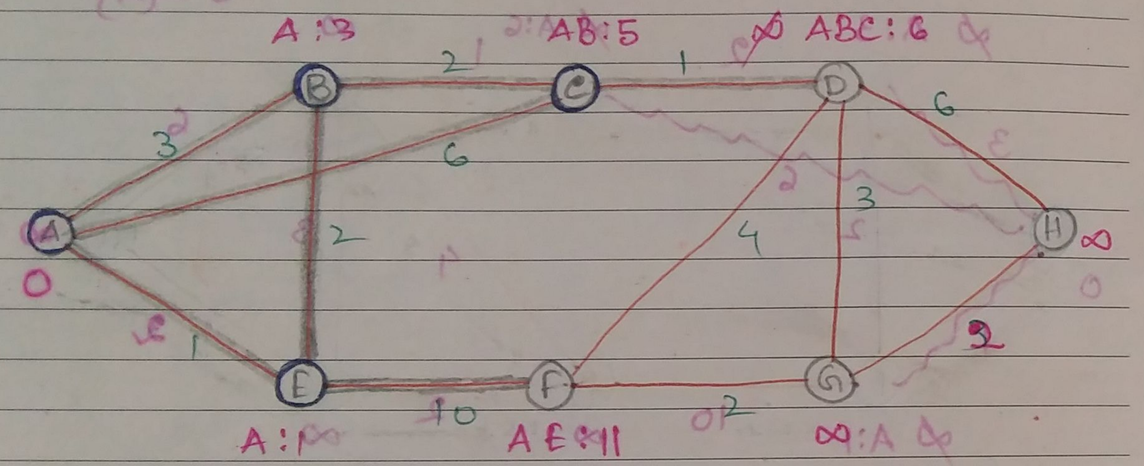

A → B → E → F → G
{7,3,8,4}=2

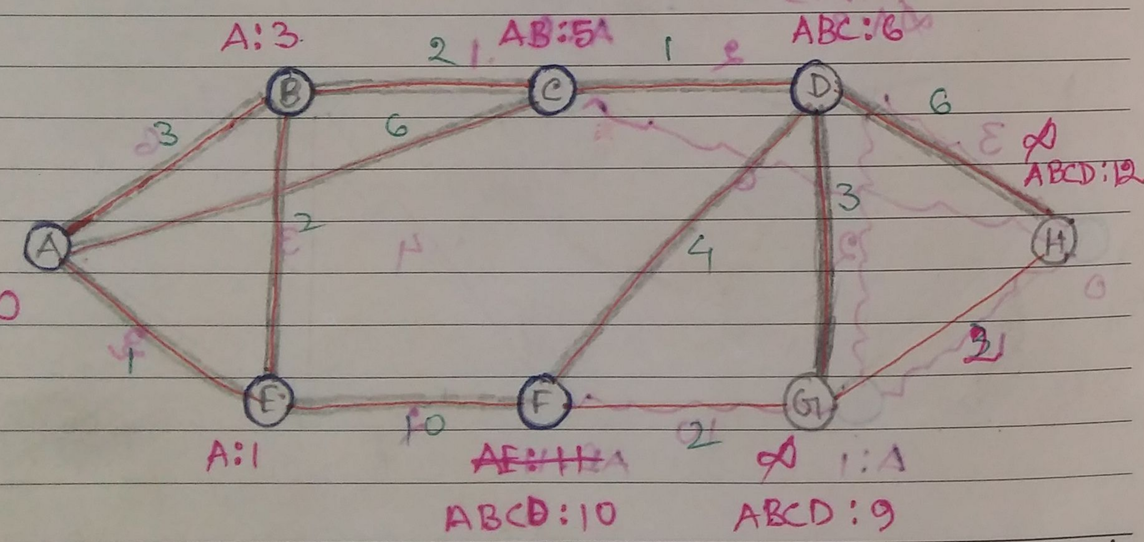


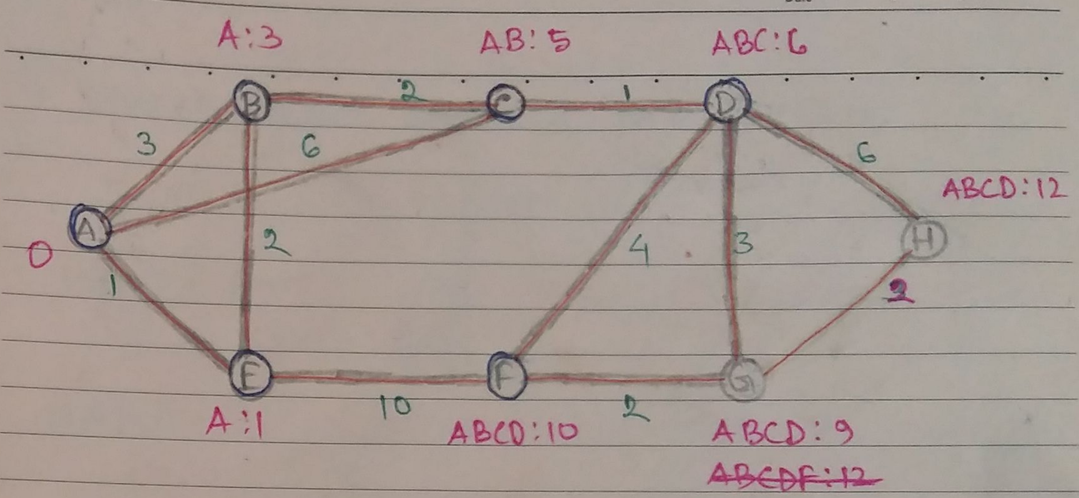

S={A}
A:3
A:6
A:1

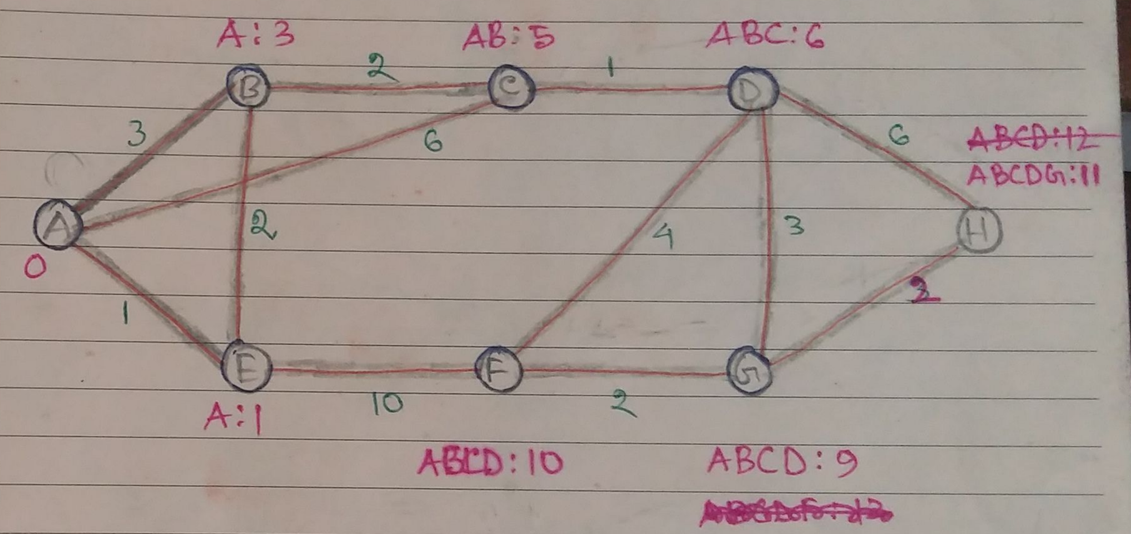

A:1











A → B → C → D → G → H

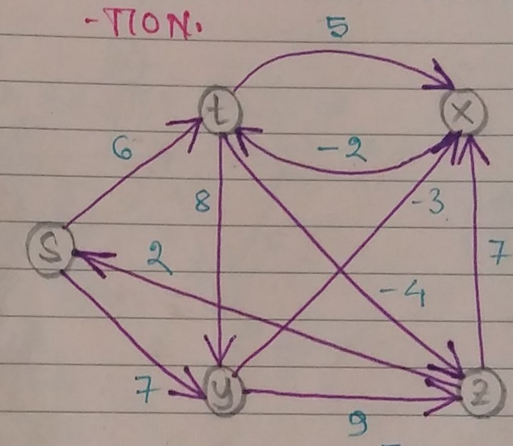
COST : 3+2+1+3+2 = 11

SINGLE SOURCE SHORTEST PATH PROBLEM:

BELLMAN-FORD ALGORITHM:

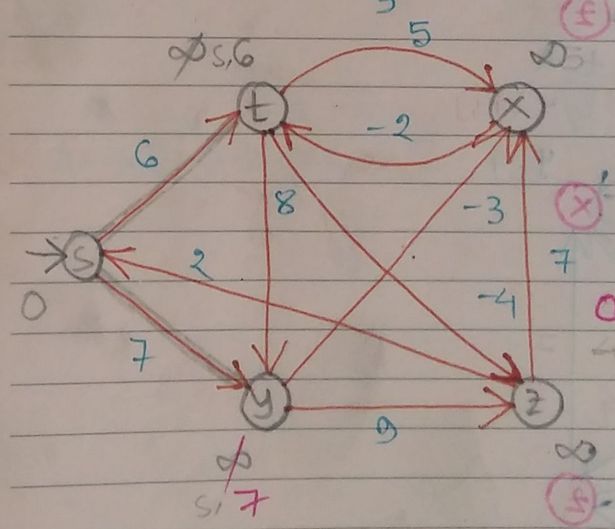
INPUT: GRAPH WITH NEGATIVE WEIGHT, A SOURCE & A DESTINATION

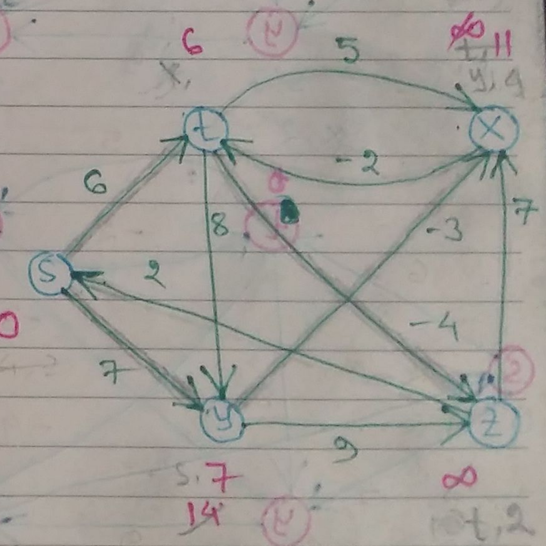
OUTPUT: SHORTEST PATH FROM THE SOURCE TO DESTINATION.

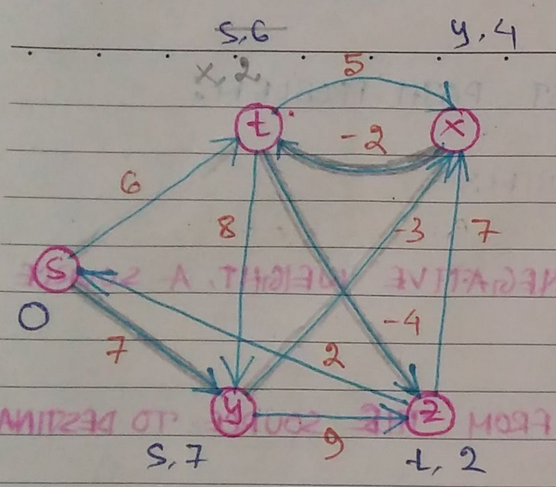


EDGE LIST:

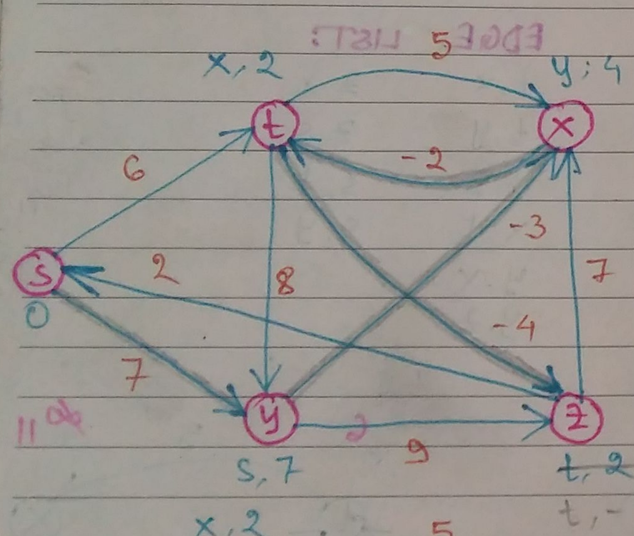
t, x	z, x
t, y	z, s
t, z	s, t
x, t	s, y
y, x	
y, z	

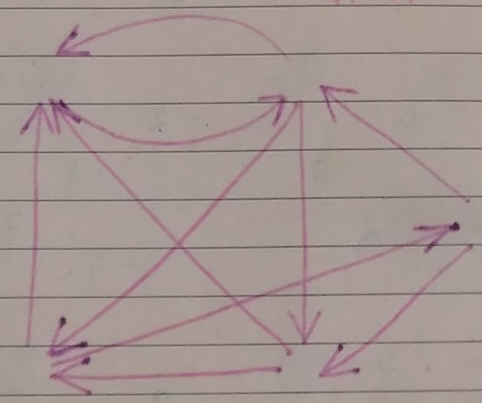


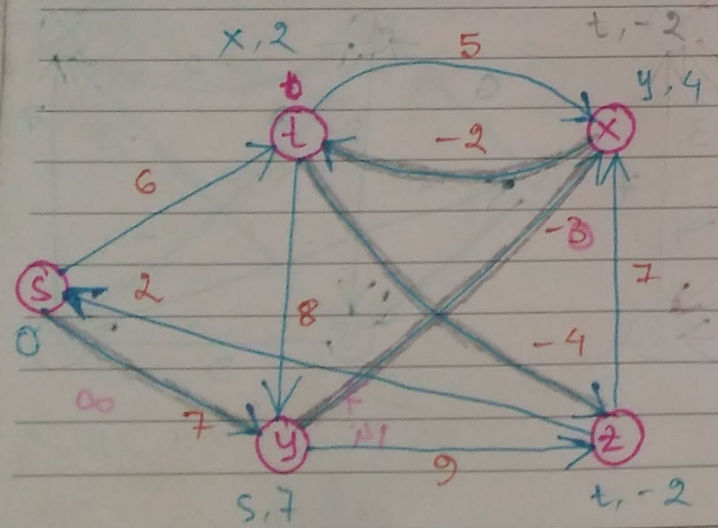


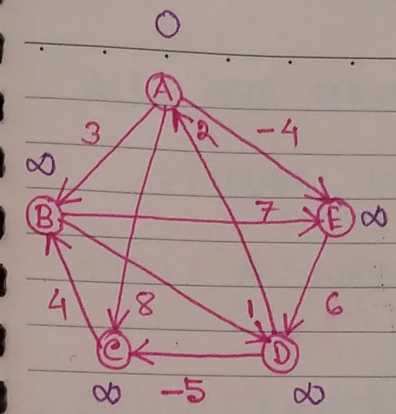


t,x	z,x
t,y	z,s
t,z	s,t,
x,t	s,y
y,x	
y,z	

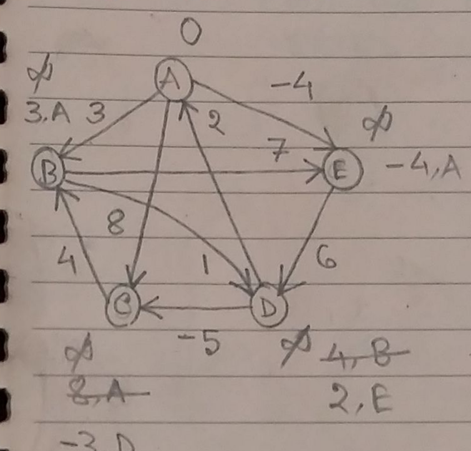


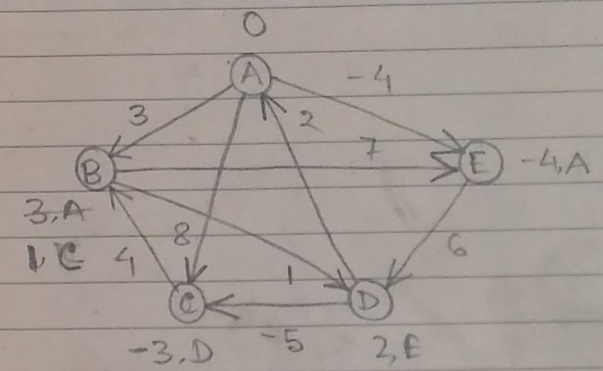


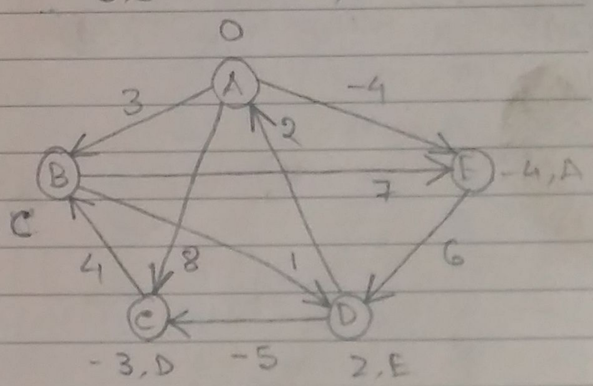




AB 3	DC -5
AC 8	DA 2
AE -4	ED 6
BD +1	BE 7
CB 4	

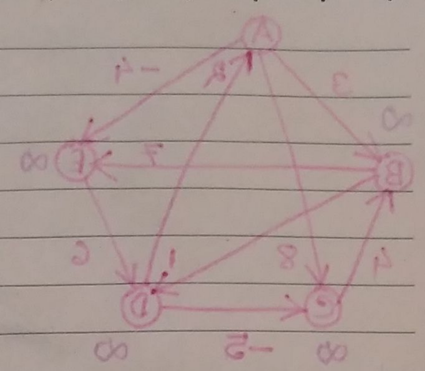






-4 6 -5 4

A → E → D → C → B 1,C

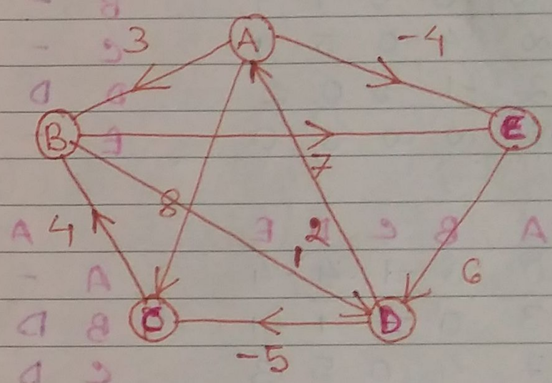


ALL PAIR SHORTEST PATH PROBLEM:

FLOYED WARSHALL ALGORITHM:

INPUT: A WEIGHTED GRAPH

OUTPUT: SHORTEST PATH BETWEEN EVERY PAIR OF VERTICS.



	A	B	C	D	E
A	0	3	8	∞	-4
B	∞	0	∞	∞	7
C	∞	4	0	∞	∞
D	2	∞	-5	0	∞
E	∞	∞	∞	-6	0

	A	B	C	D	E
A	-	A	A	-	A
B	-	-	-	B	B
C	-	C	-	-	-
D	D	-	D	-	-
E	-	-	-	E	-

	A	B	C	D	E
A	0	3	8	4	-4
B	∞	0	∞	1	7
C	∞	4	0	5	11
D	2	5	-5	0	-2
E	∞	∞	∞	6	0

	A	B	C	D	E
A	-	A	A	B	A
B	-	-	-	B	B
C	-	C	-	B	B
D	D	A	D	-	A
E	-	-	-	E	-

	A	B	C	D	E
A	0	3	8	4	-4
B	∞	0	∞	1	7
C	∞	4	0	5	11
D	2	-1	-5	0	2
E	∞	∞	∞	6	0

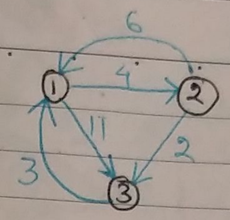
	A	B	C	D	E
A	-	A	A	B	A
B	-	-	-	B	B
C	-	C	-	B	B
D	D	C	D	-	A
E	-	-	-	E	-

	A	B	C	D	E
A	0	3	-1	4	-4
B	3	0	-4	1	-1
C	7	4	0	5	3
D	2	-1	-5	0	-2
E	8	5	1	6	0

	A	B	C	D	E
A	-	A	D	B	A
B	D	-	D	B	D
C	D	C	-	B	A
D	D	C	D	-	A
E	D	C	D	E	-

	A	B	C	D	E
A	0	1	-3	-2	4
B	3	0	-4	-1	1
C	7	4	0	5	3
D	2	-1	-5	0	-2
E	8	5	1	6	0

	A	B	C	D	E
A	-	A	D	B	A
B	D	-	D	B	D
C	D	C	-	B	A
D	D	C	D	-	A
E	D	C	D	E	-



$A^0 =$

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

	1	2	3
1	-	1	1
2	2	-	2
3	3	-	-

$A1 =$

	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

	1	2	3
1	-	1	1
2	2	-	2
3	3	1	-

$A2 =$

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

	1	2	3
1	-	1	2
2	2	-	2
3	3	1	-

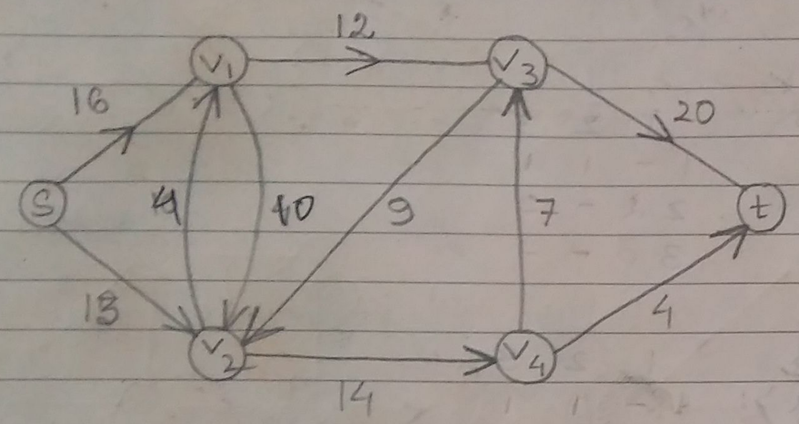
$A3 =$

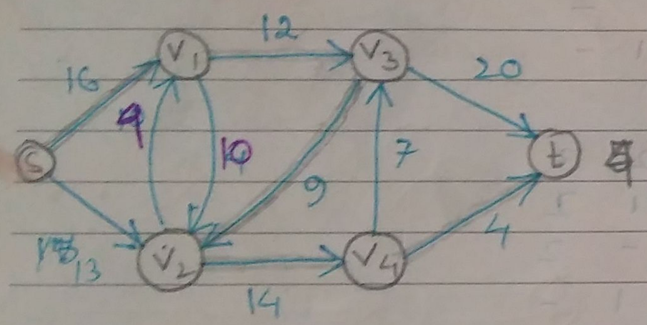
	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

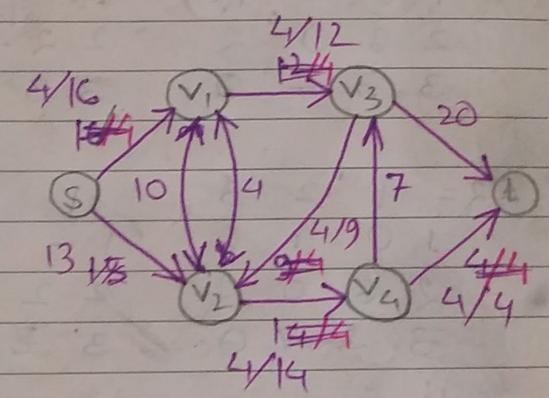
	1	2	3
1	-	1	2
2	3	-	2
3	3	1	-

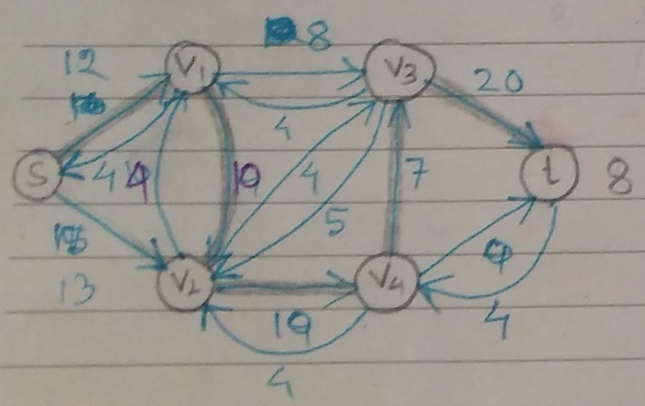
FLOW NETWORK:

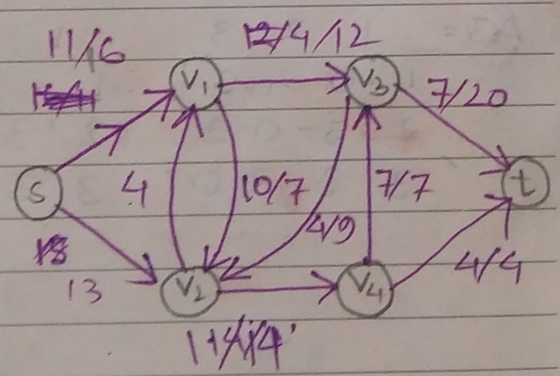
FORD FULKERSON ALGORITHM:

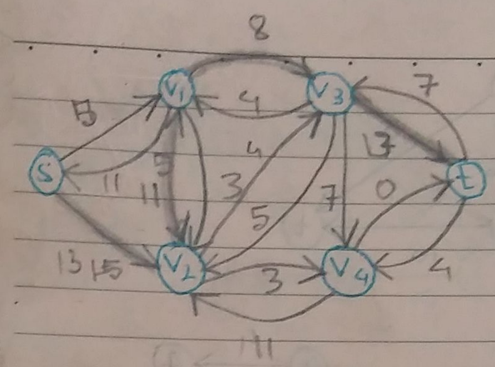


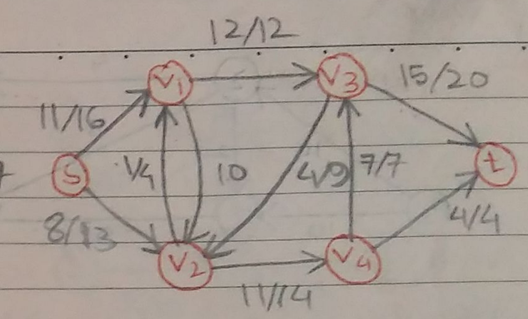


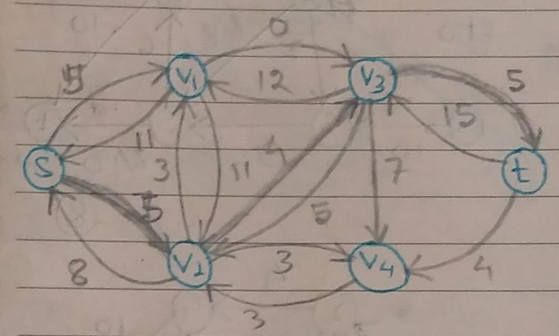


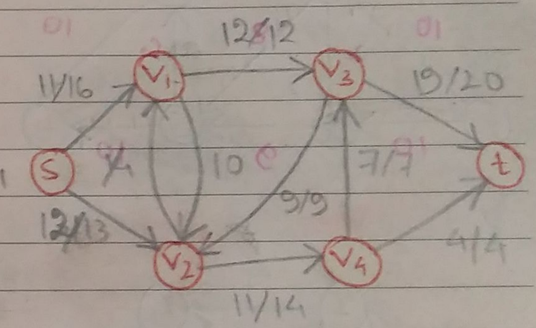


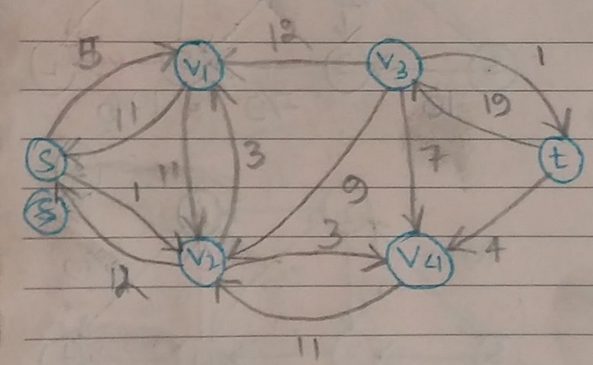


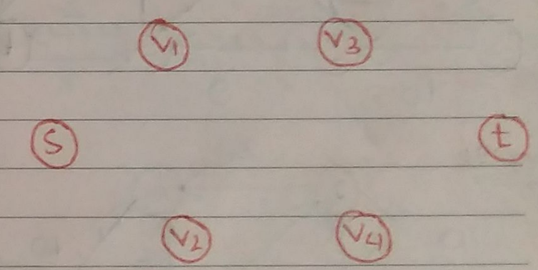






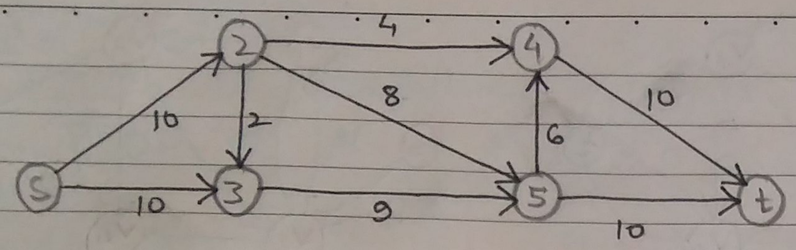


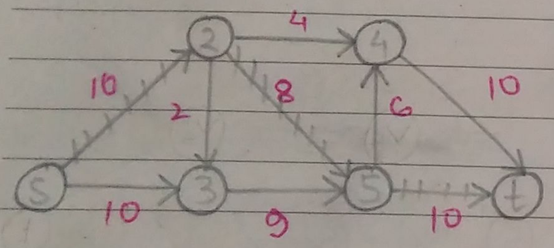


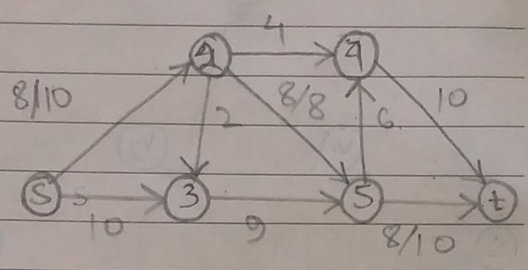


A2M SUPREME

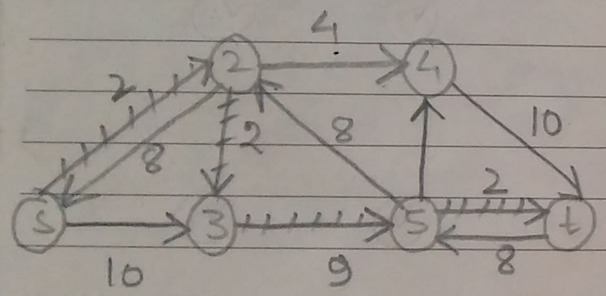
A2M SUPREME

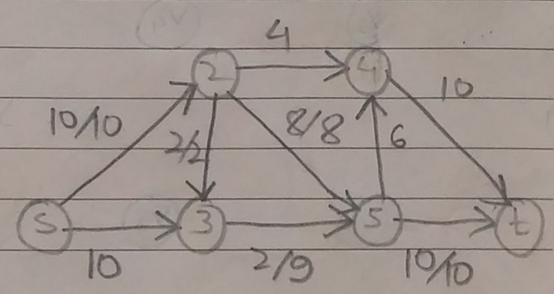




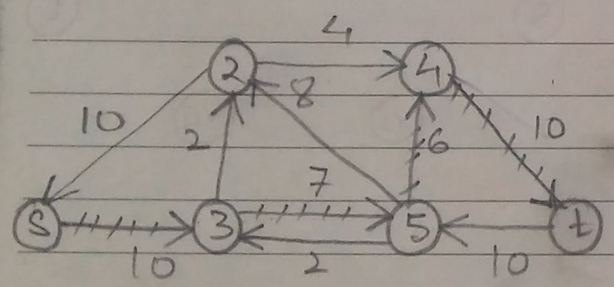


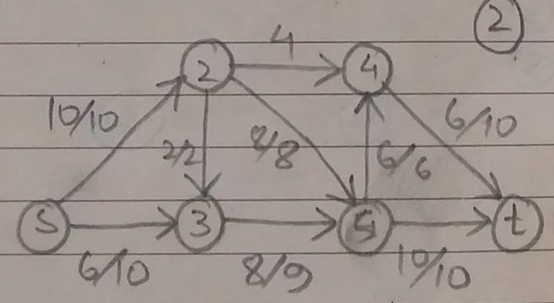
8



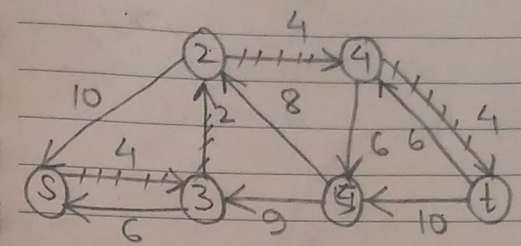


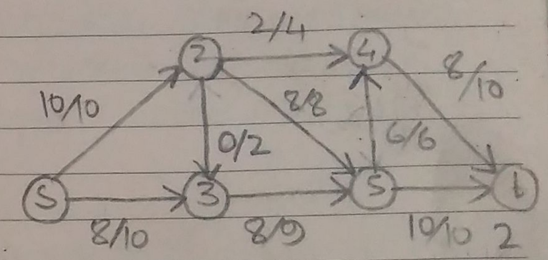
2

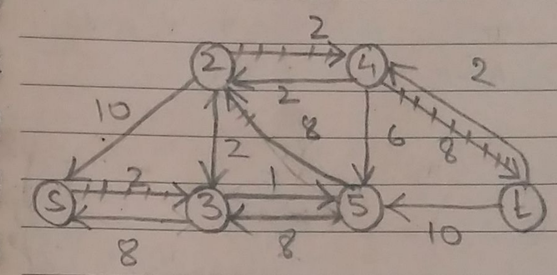


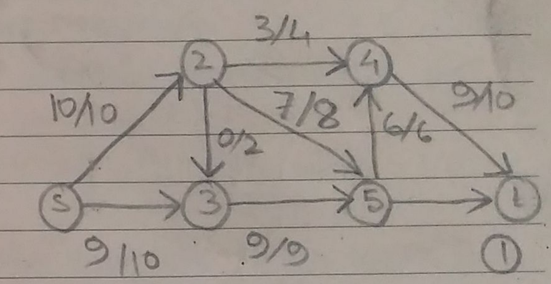


6

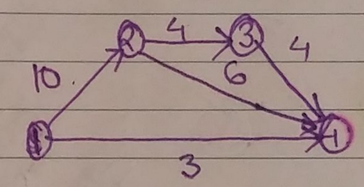








1



No.

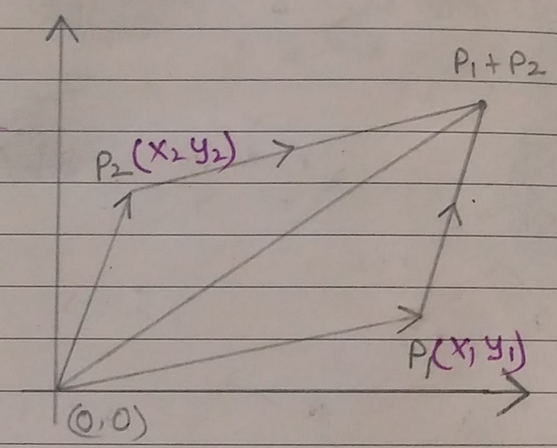
Date

COMPUTATIONAL GEOMETRY: is A BRANCH OF COMPUTER SCIENCE DEVOTED TO THE STUDY OF ALGORITHM WHICH CAN BE STATED IN TERMS OF GEOMETRY.

LINE SEGMENT: is A PART OF A LINE THAT IS BOUNDED BY TWO DISTINCT END POINTS. AND CONTAINS EVERY POINT ON THE LINE BETWEEN ITS END POINTS.

LINE SEGMENT INTERSECTION:

$P_1 \times P_2$ = cross product
= signed area of the parallelogram



$$P_1 \times P_2 = \det \begin{pmatrix} x_1 & x_2 \\ y_1 & y_2 \end{pmatrix}$$

$$= x_1 y_2 - x_2 y_1$$

$$=$$

If $P_1 \times P_2$ POSITIVE P_1 is clockwise from P_2, origin
NEGATIVE " " counter-clockwise " " "
ZERO P_1 & P_2 collinear " "

A2M SUPREME

No.

Date

Respect to Point P_0 transform to origin

$\overrightarrow{(P_0 - P_1)}$ is cw from $\overrightarrow{(P_0 P_2)}$, origin transform to P_0

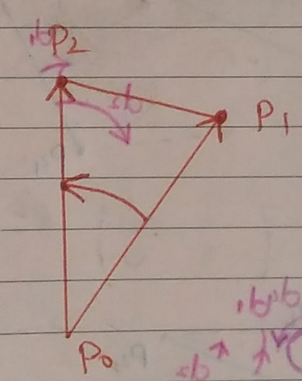
$$(P_1 - P_0) = P_1' = (x_1' y_1') = (x_1 - x_0, y_1 - y_0)$$

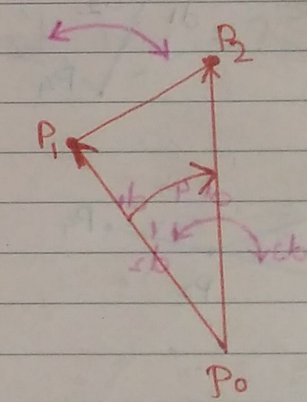
$$(P_2 - P_0) = P_2' = (x_2' y_2') = (x_2 - x_0, y_2 - y_0)$$

CROSS PRODUCT:

$$(P_1 - P_0) \times (P_2 - P_0) = (x_1 - x_0, y_1 - y_0) \times (x_2 - x_0, y_2 - y_0)$$

if + $\overrightarrow{P_0 P_1}$ is cw from $\overrightarrow{P_0 P_2}$
" " ccw " "
0 " " colinear with " "





A2M SUPREME

FOR 3 consecutive points.

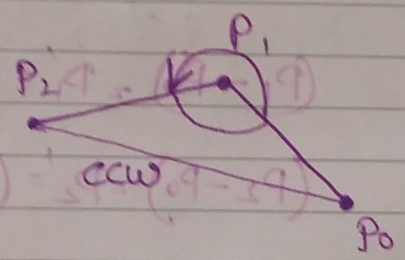
$\overline{P_0P_1}$, $\overline{P_1P_2}$, TURN LEFT/RIGHT AT P_1

$(P_2 - P_0) \times (P_1 - P_0)$

$-$ $\overline{P_0P_2}$ ccw P_0P_1

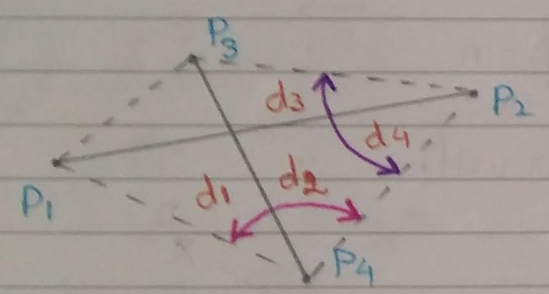
$+$ " cw "

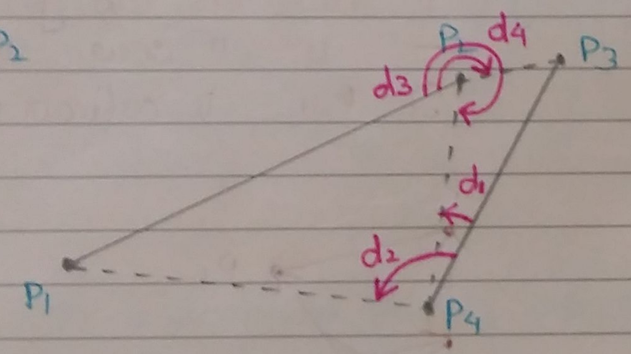
0 " collinear " "

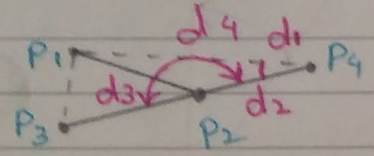


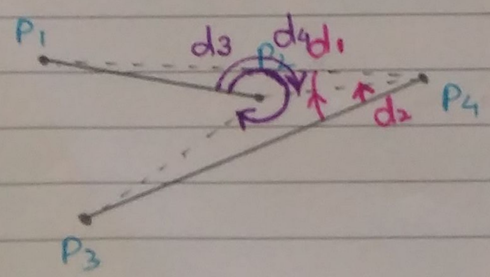
LINE SEGMENT INTERSECTION:

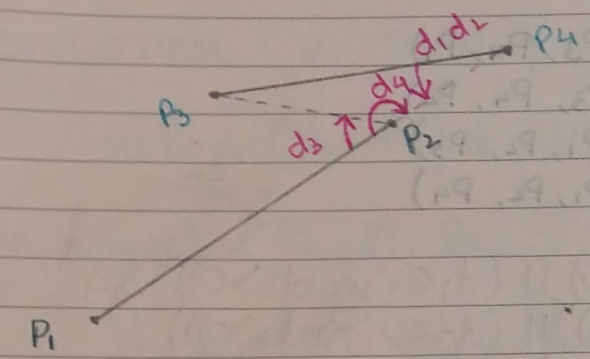
LINE SEGMENT: $\overline{P_1P_2}$, $\overline{P_3P_4}$

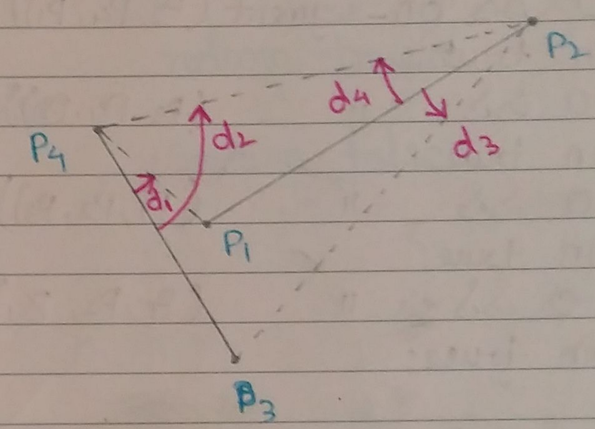










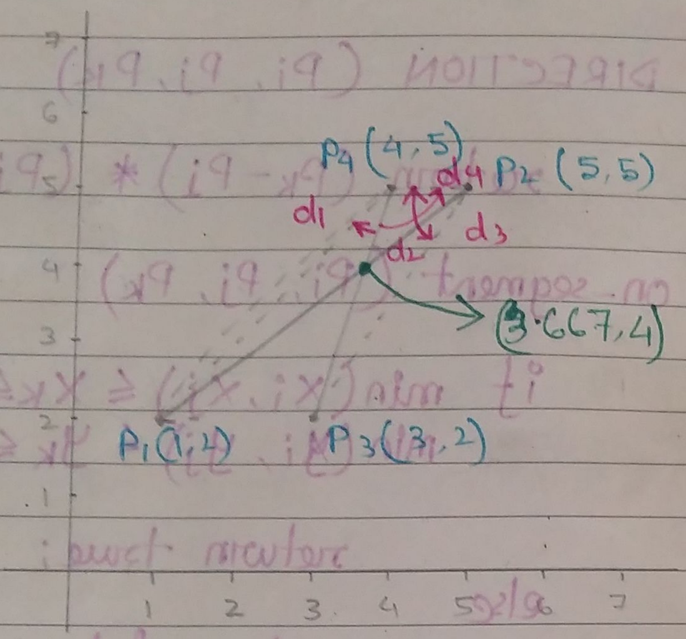


d_1 = direction (P_3, P_4, P_1)
d_2 = " (P_3, P_4, P_2)
d_3 = " (P_1, P_2, P_3)
d_4 = " (P_1, P_2, P_4)

```
if (((d1>0 && d2<0) || (d1<0 && d2>0)) &&
    ((d3>0 && d4<0) || (d3<0 && d4>0)))
        return true;
else if (d1=0 && on-segment (P3, P4, P1))
        return true;
else if (d2=0 && "  (P3, P4, P2))
        return true;
else if (d3=0 && "  (P1, P2, P3))
        return true;
else if (d4=0 && "  (P1, P2, P4))
        return true;
else
        return false;
```



```
DIRECTION (Pi, Pj, Pk)
    return (Pk-Pi) * (Pj-Pi);

on-segment (Pi, Pj, Pk)
    if min(xi,xj) ≤ xk ≤ max(xi,xj) &&
       min(yi,yj) ≤ yk ≤ max(yi,yj)
            return true;
    else
            return false;
```



d_1 = direction $(P_3, P_4, P_1) = (3-1, 2-2) * (4-1, 5-2)$

d_2 = " $(P_3, P_4, P_2) = (3-5, 2-5) * (4-5, 5-5)$

d_3 = " $(P_1, P_2, P_3) = (1-3, 2-2) * (5-3, 5-2)$

d_4 = " $(P_1, P_2, P_4) = (1-4, 2-5) * (5-4, 5-5)$

$d_1 = (2, 0) \times (3, 3) = \begin{pmatrix} 2 & 3 \\ 0 & 3 \end{pmatrix} = 6$

$d_3 = \begin{pmatrix} -2 & 2 \\ 0 & +3 \end{pmatrix} = -6$

$d_2 = \begin{pmatrix} -2 & -1 \\ -3 & 0 \end{pmatrix} = -3$

$d_4 = \begin{pmatrix} -3 & 1 \\ -3 & 0 \end{pmatrix} = 3$

$\frac{x_1 - x_2}{x_2 - x} = \frac{y_1 - y_2}{y_2 - y}$ LINE EQUATION

$P_1 P_2$ = EQUATION

$\Rightarrow \frac{5-1}{1-x} = \frac{5-2}{2-y}$

$\Rightarrow 4(2-y) = 3(1-x)$

$\Rightarrow 3 - 3x = 8 - 4y$

$\Rightarrow -3x + 4y - 5 = 0$

$\Rightarrow 3x - 4y + 5 = 0$ — (I)

$P_3 P_4$ EQUATON

$\frac{4-3}{3-x} = \frac{5-2}{2-y}$

$\Rightarrow 1(2-y) = 3(3-x)$

$\Rightarrow 2 - y = 9 - 3x$

$\Rightarrow 3x - y - 7 = 0$

$\Rightarrow 3x - y - 7 = 0$ — (II)

(I) − (II)

$$\begin{array}{r} 3x - 4y + 5 = 0 \\ 3x - y - 7 = 0 \\ \hline -3y + 12 = 0 \end{array}$$

$\Rightarrow y = 4$

$3 \cdot x - 4 - 7 = 0$

$\Rightarrow 3x = 11$

$\therefore x = 11/3 = 3.667$

$(P_3, P_4, P_1) = (3-4, 2-5) * (1-4, 2-5) =$
$= (-1, -3) * (-3, -3) = 4 - 9 = -5$

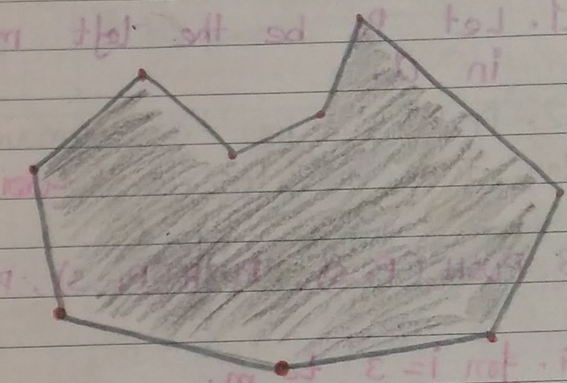
$(P_3, P_4, P_2) = (3-5, 2-5) * (5-4, 5-5)$
$= (-1, -3) * (1, 0) = -1 + 3 = 2$

$(P_1, P_2, P_3) = (1-5, 2-5) * (3-5, 2-5) =$
$= (-4, -3) * (-2, -3) = 8 - 9 = -1$

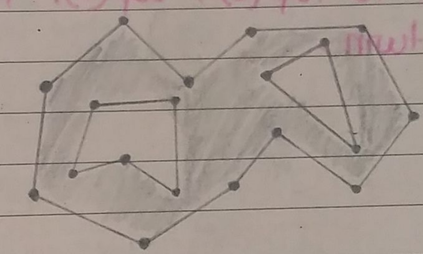
$(P_1, P_2, P_4) = (-4, -3) * (-1, 0) = +4 + 0 = 4$

POLYGON: A CONNECTED REGION OF THE PLANE BOUNDED BY A SEQUENCE OF LINE SEGMENT.

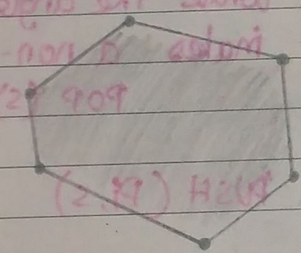
i) SIMPLE POLYGON:



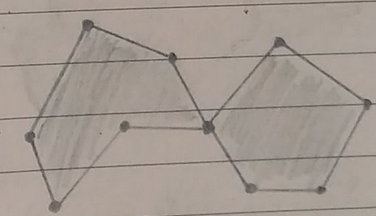
ii) POLYGON WITH HOLES:



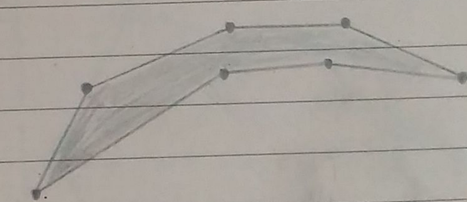
iii) CONVEX POLYGON:



iv) NON-SIMPLE POLYGON:



v) CONCAVE POLYGON:



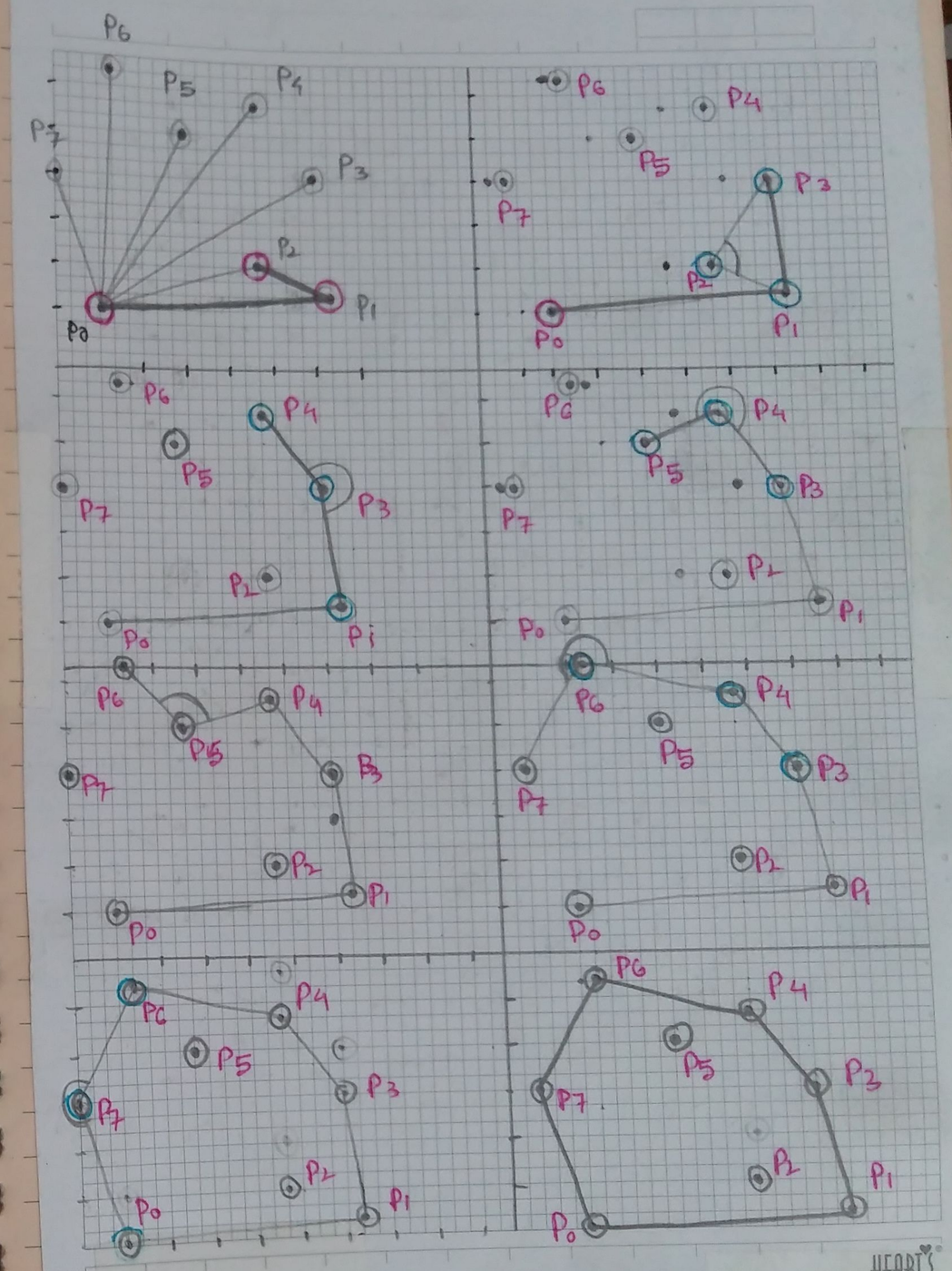
CONVEX POLYGON FINDING: GRAHAM SCAN ALGORITHM

INPUT: A SET OF POINTS.
OUTPUT: A SET OF POINTS CONTAINING CONVEX HULL.

1. Let P_0 be the left most/minimum Y coordinate in Q.
2. P_1, P_2, P_m be the remaining points sorted by polar angle in the cw order around P_0.
3. PUSH (P_0, S), PUSH (P_1, S), PUSH (P_2, S)
4. for i = 3 to m.

   while the angle (next-to-top(S), top(S), P_i) makes a non-left turn,
   POP (S);

   PUSH (P_i, S)
5. return S.



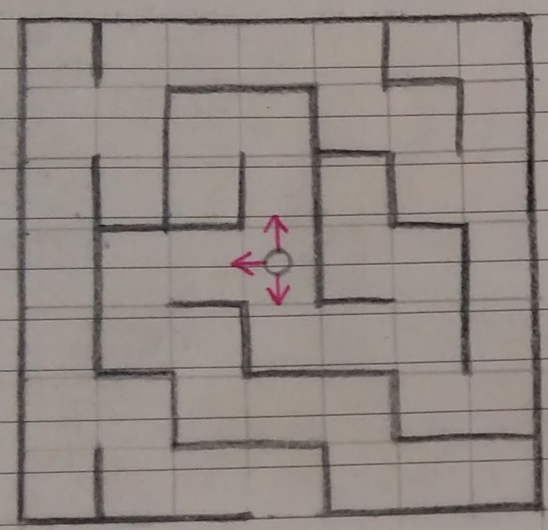
No.

Date

BACKTRACKING:

AN ALGORITHM FOR SYSTEMATIC SEARCH THROUGH ALL THE POSSIBLE COMBINATIONS WITHIN A SEARCH SPACE.

INCREMENTALLY ATTEMPTS TO EXTEND A PARTIAL SOLUTION TOWARD A COMPLETE SOLUTION.



We explore each node as follows

→ to explore node N
if N is a goal state, return success
" " " " leaf node, " failure

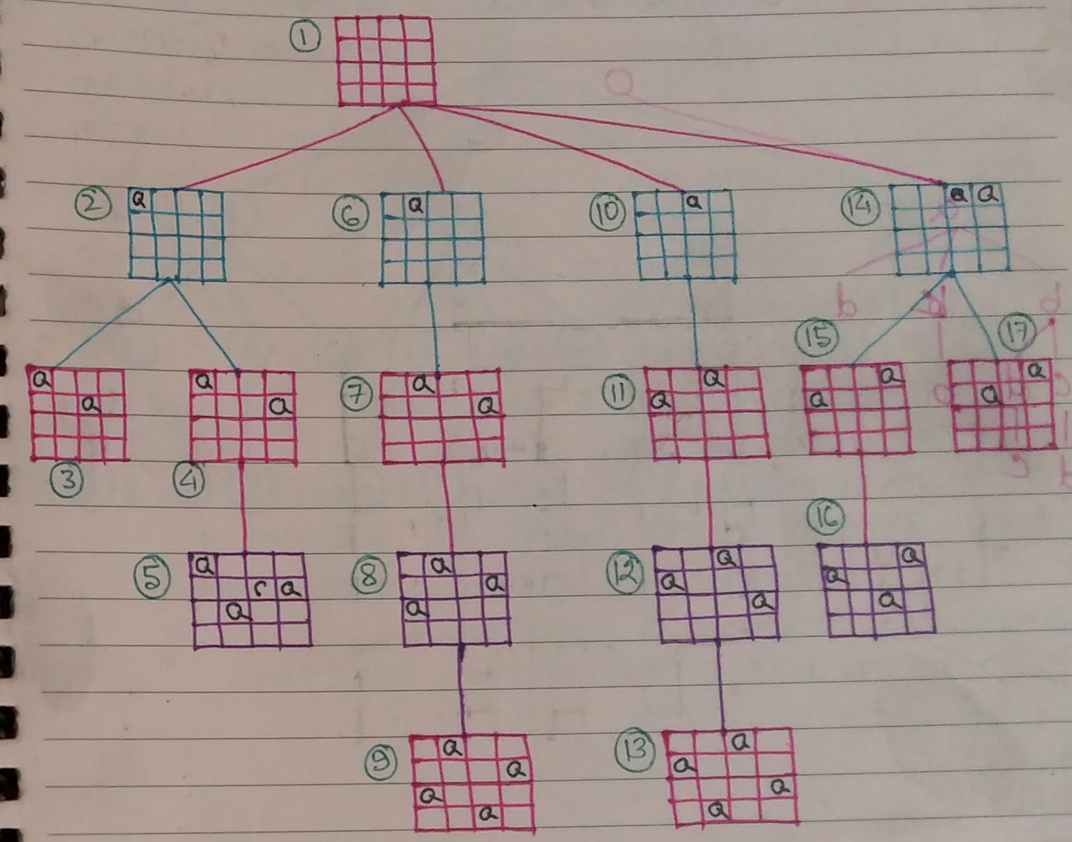
for each child c of N:
Explore c.
if c is successful, return success
return failure.

A2M SUPREME

No.

Date

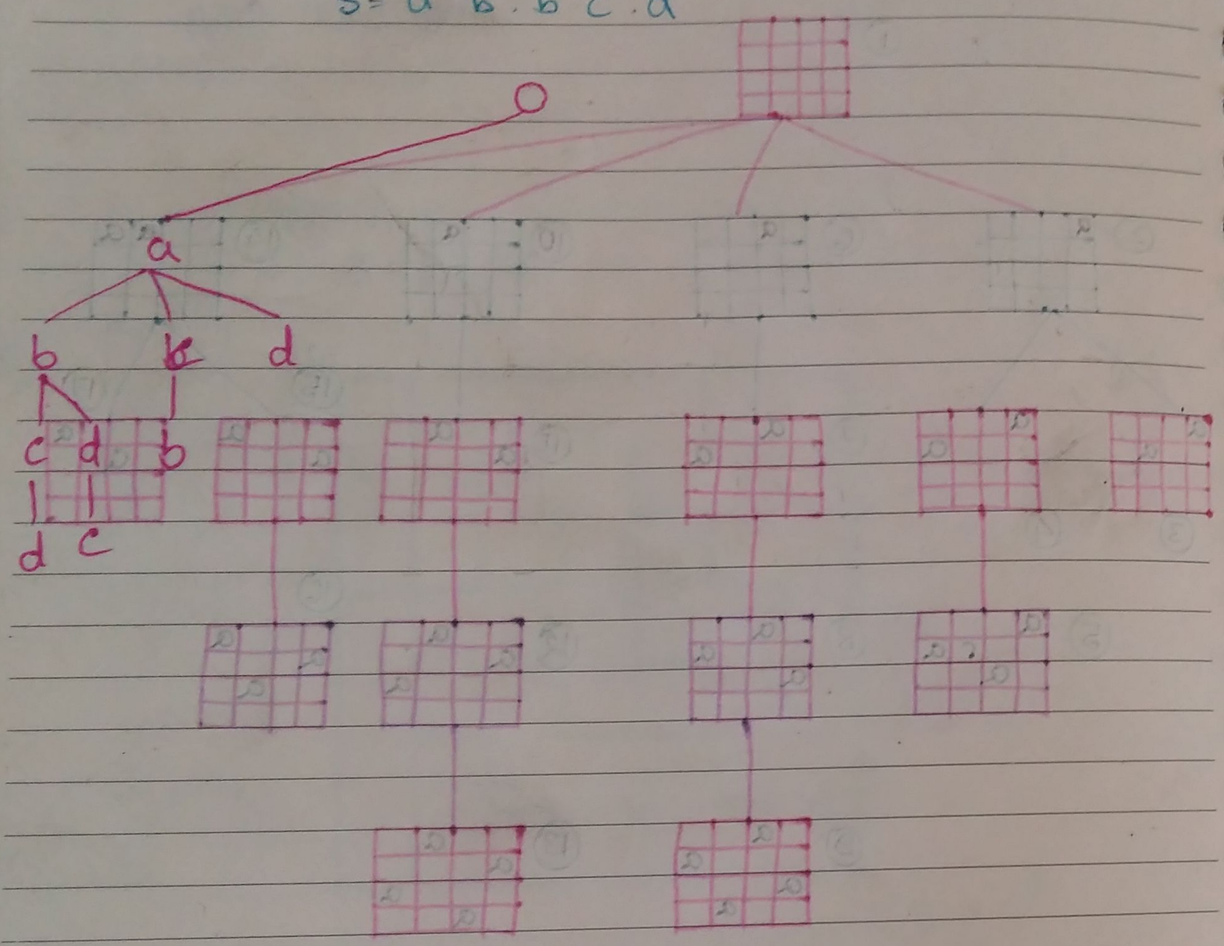
N-QUEEN PROBLEM



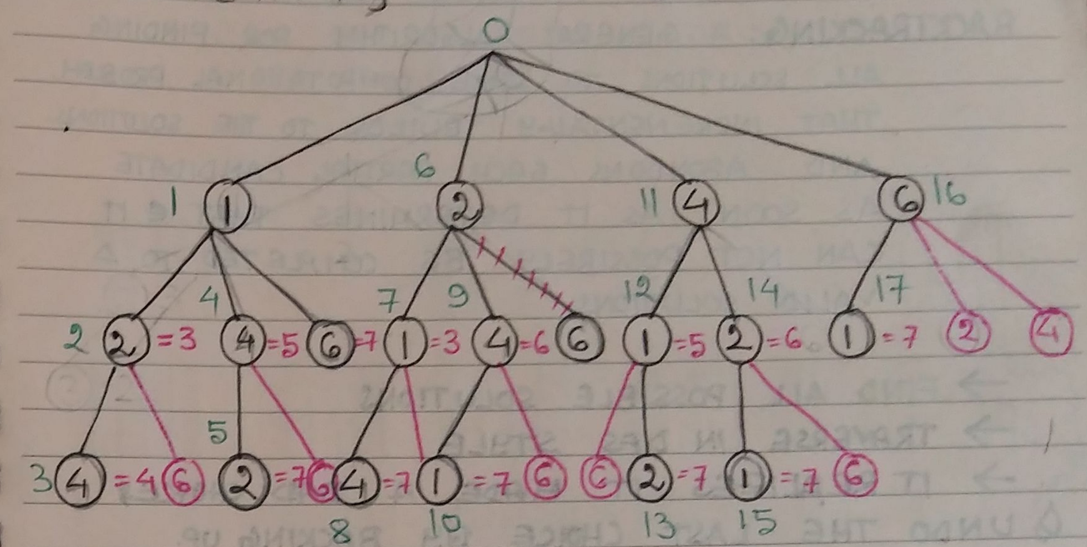
A2M SUPREME

ALL PERMUTATION:

S = 'a' 'b' 'b' 'c' 'd'



S = {1, 2, 4, 6} SUM = 7



No.

Date

BACKTRACKING: A GENERAL ALGORITHM FOR FINDING ALL SOLUTIONS TO SOME COMPUTATIONAL PROBLEM, THAT INCREMENTALLY BUILDS TO THE SOLUTIONS AND ABONDONS EACH PARTIAL CANDIDATE AS SOON AS IT DETERMINES THAT IT CAN NOT POSSIBELY BE COMPLETED TO A VALID SOLUTIONS.

→ FIND ALL POSSIBLE SOLUTIONS
→ TRAVERSE IN DFS STYLE
→ IT REALIZES IT MADE A BAD CHOICE & UNDO THE LAST CHOICE BY BACKING UP.
→ IT SEARCH THE STATE SPACE TREE UNTIL IT FOUND A SOLUTION.
→ IT INVOLVES FEASIBILITY FUNCTIONS.

No.

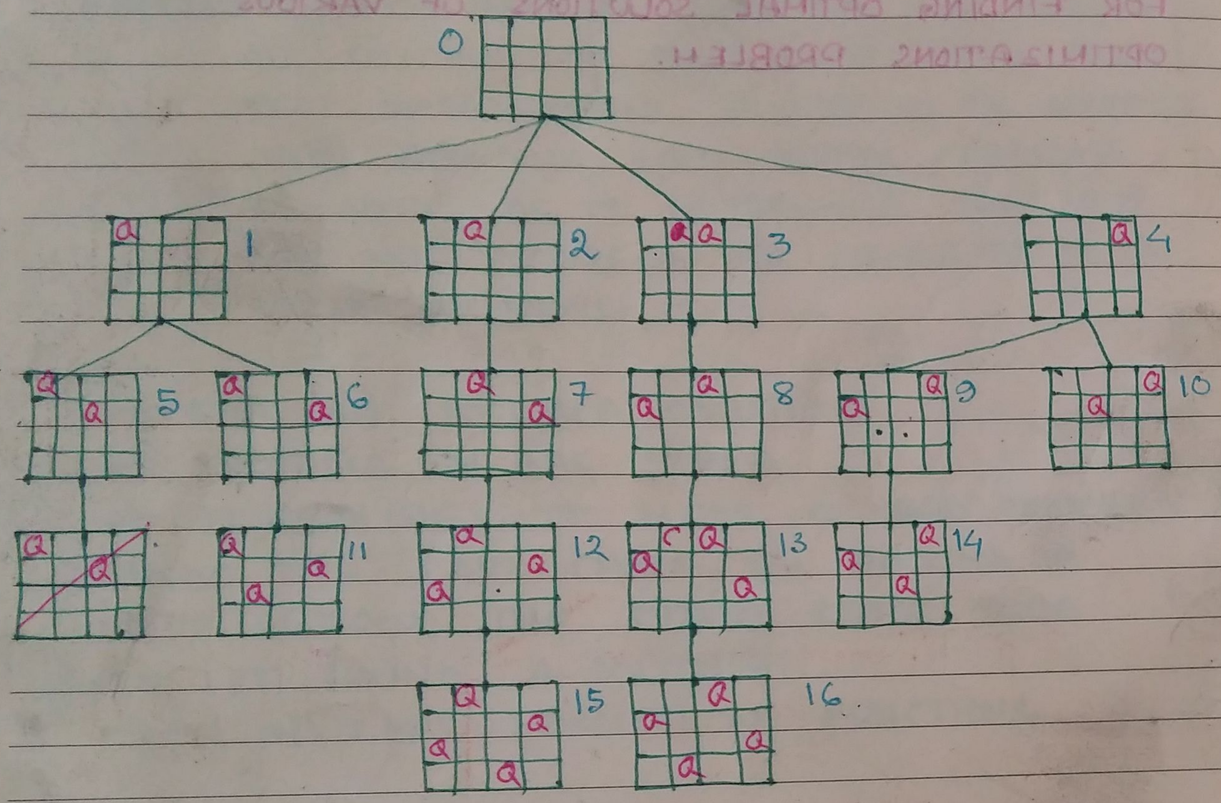
Date

BRANCH AND BOUND: A GENERAL ALGORITHM FOR FINDING OPTIMAL SOLUTIONS OF VARIOUS OPTIMIZATIONS PROBLEM.

No.

Date

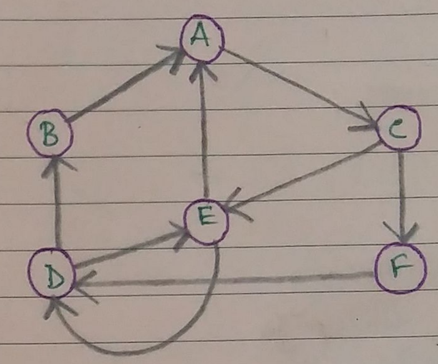
n-queen problem



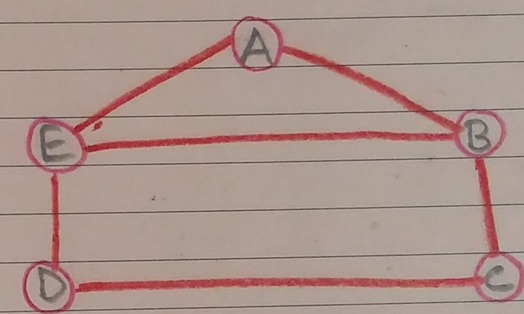
No.

Date

EULAR PATH:



C E D B A C F D E A



E A B E D C B
E B A E D C B
B A E B C D E
B E A B C D E

No.

Date

ELEMENTARY NUMBER THEORY:

$Z = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ INTEGER NUMBER

$N = \{0, 1, 2, 3, \ldots\}$ NATURAL NUMBER

$d \mid a$ = d divides a

PRIME NUMBER: $a > 1$ & only divisers are 1 & a.

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 79, 83, 87, 89, 97

COMPOSITE NUMBER: IF NOT PRIME, THEN COMPOSITE

$a = qn + r$ q = quotient

$r = a \bmod n$ r = remainder

G.C.D: GREATEST COMMON DIVISOR

largest common divisor of a & b, gcd (a,b)

GCD(24,30) = 6

GCD(5,7) = 1

GCD(0,9) = 9

if (GCD(17,5) = 1) RELATIVE PRIME

A2M SUPREME

No.

Date

UNIQUE FACTORIZATION: AN INTEGER, a

$$a = p_1^{e_1} * p_2^{e_2} * p_3^{e_3} * \cdots * p_n^{e_n}$$

p_i, $p_1 < p_2 < \cdots < p_n$, p_i = Prime

e_i = POSITIVE NUMBER

$$6000 = 2^4 \cdot 3 \cdot 5^3$$

EUCLID'S ALGORITHM:

```
euclid (a,b)
{
    if (b==0)
        then return a;
    else
        return euclid (b, a%b)
}
```

EUCLID (30,21) = EUCLID(21,9)

= EUCLID(9,3)

= EUCLID (3,0)

= 3

A2M SUPR

No.

Date

SEARCHING:

ALGORITHM	Data Structure	TIME COMPLEXITY Average	TIME COMPLEXITY Worst	SPACE COMPLEXITY Worst
DFS	$\|V\|, \|E\|$	–	$O(\|E\|+\|V\|)$	$O(\|V\|)$
BFS	"	–	$O(\|E\|+\|V\|)$	"
BINARY SEARCH		$O(\log n)$	$O(\log n)$	$O(1)$
LINEAR "		$O(n)$	$O(n)$	$O(1)$
DIJKSTRA'S, MIN HEAP	$\|V\|, \|E\|$	$O((\|V\|+\|E\|)\log\|V\|)$ →		$O(\|V\|)$
" , UNSORTED	"	$O(\|V\|^2)$	$O(\|V\|^2)$	"
BELLMAN-FORD	"	$O(\|V\|\|E\|)$	"	"

	Best	Average	Worst
QUICK SORT	$O(n\log n)$	$O(n\log n)$	$O(n^2)$
MERGE "	"	"	$O(n\log n)$
HEAP "	"	"	"
BUBBLE "	$O(n)$	$O(n^2)$	$O(n^2)$
INSERTION "	"	"	"
SELECTION "	$O(n^2)$	"	"
BUCKET "	$O(n+k)$	$O(n+k)$	"

No.

Date

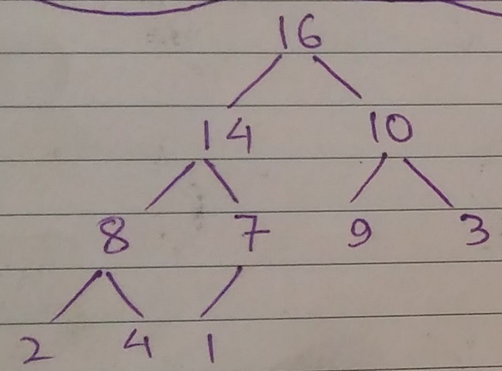
V) HEAP SORT:

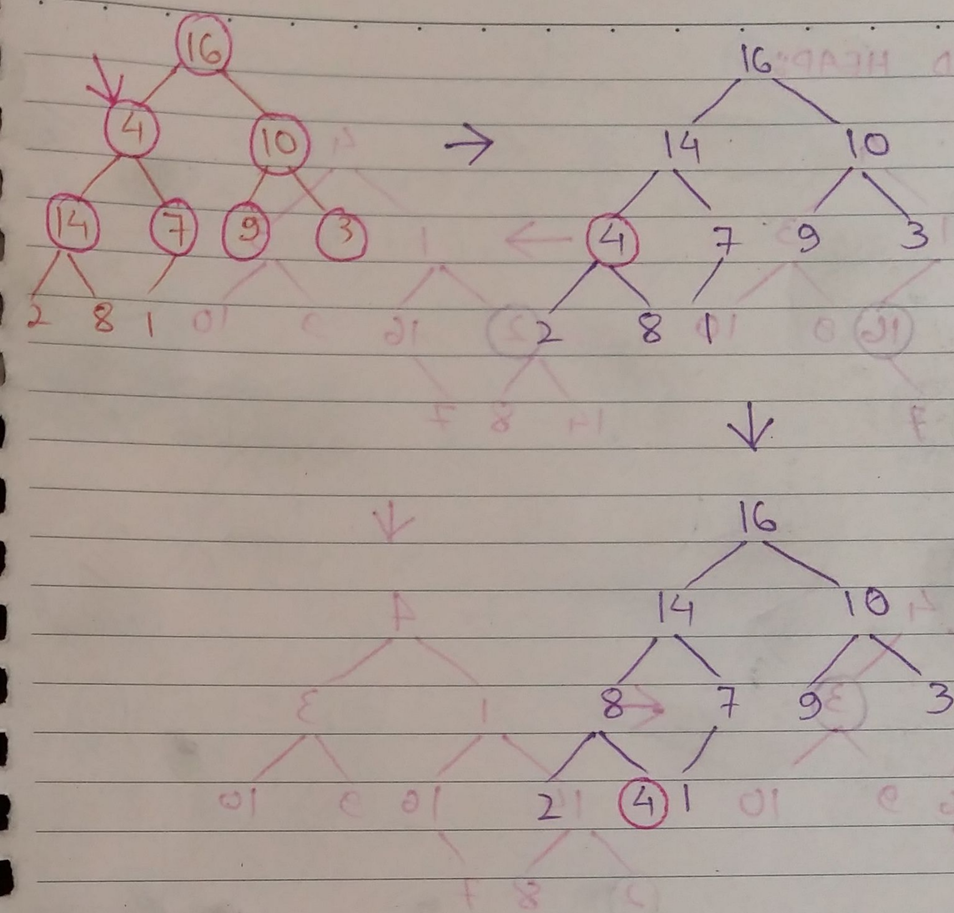
HEAP PROPERTY: A [Parent (i)] ≥ A[i], max

A [Parent (i)] ≤ A[i], min

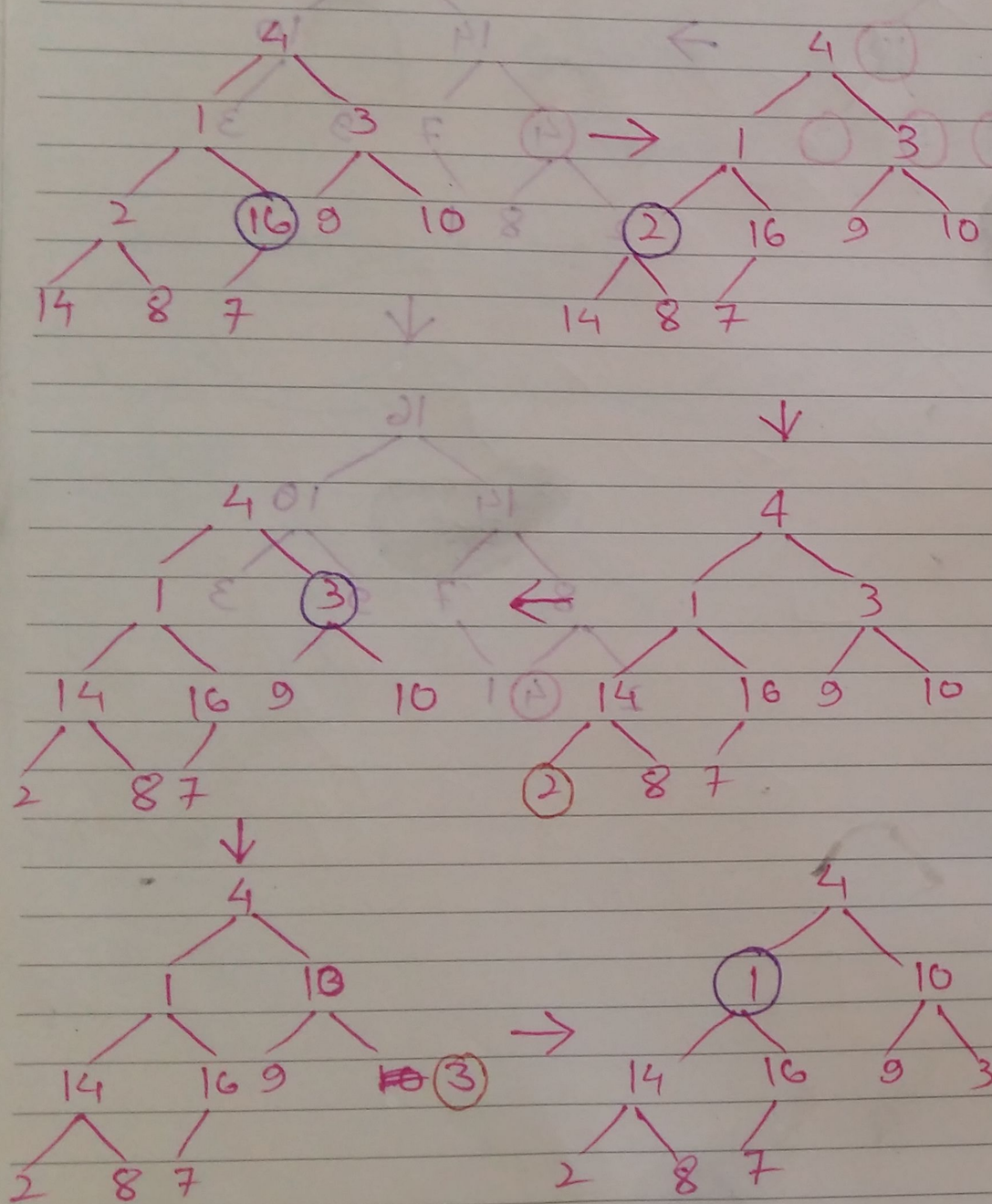
max heap:

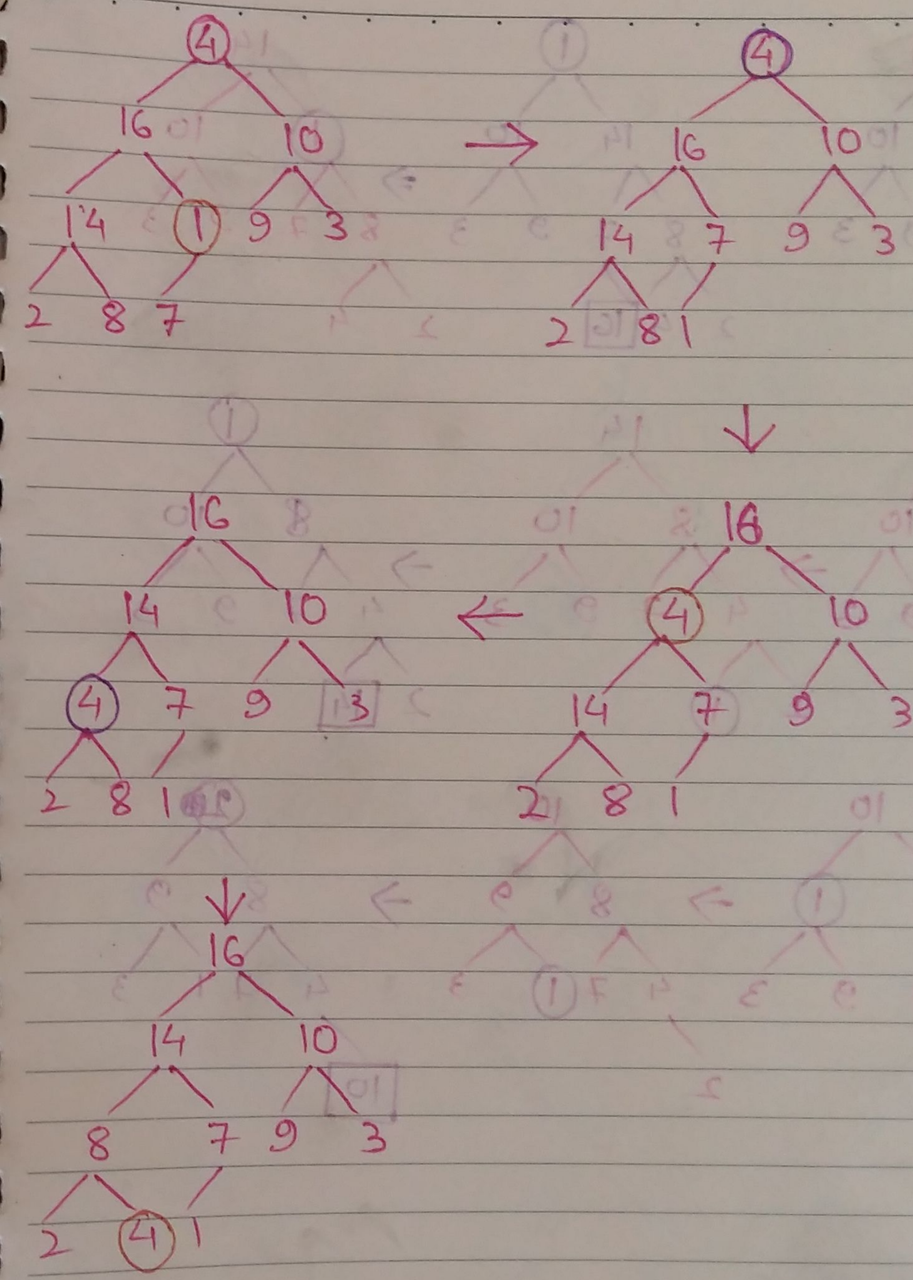
16 14 10 8 7 9 3 2 4 1

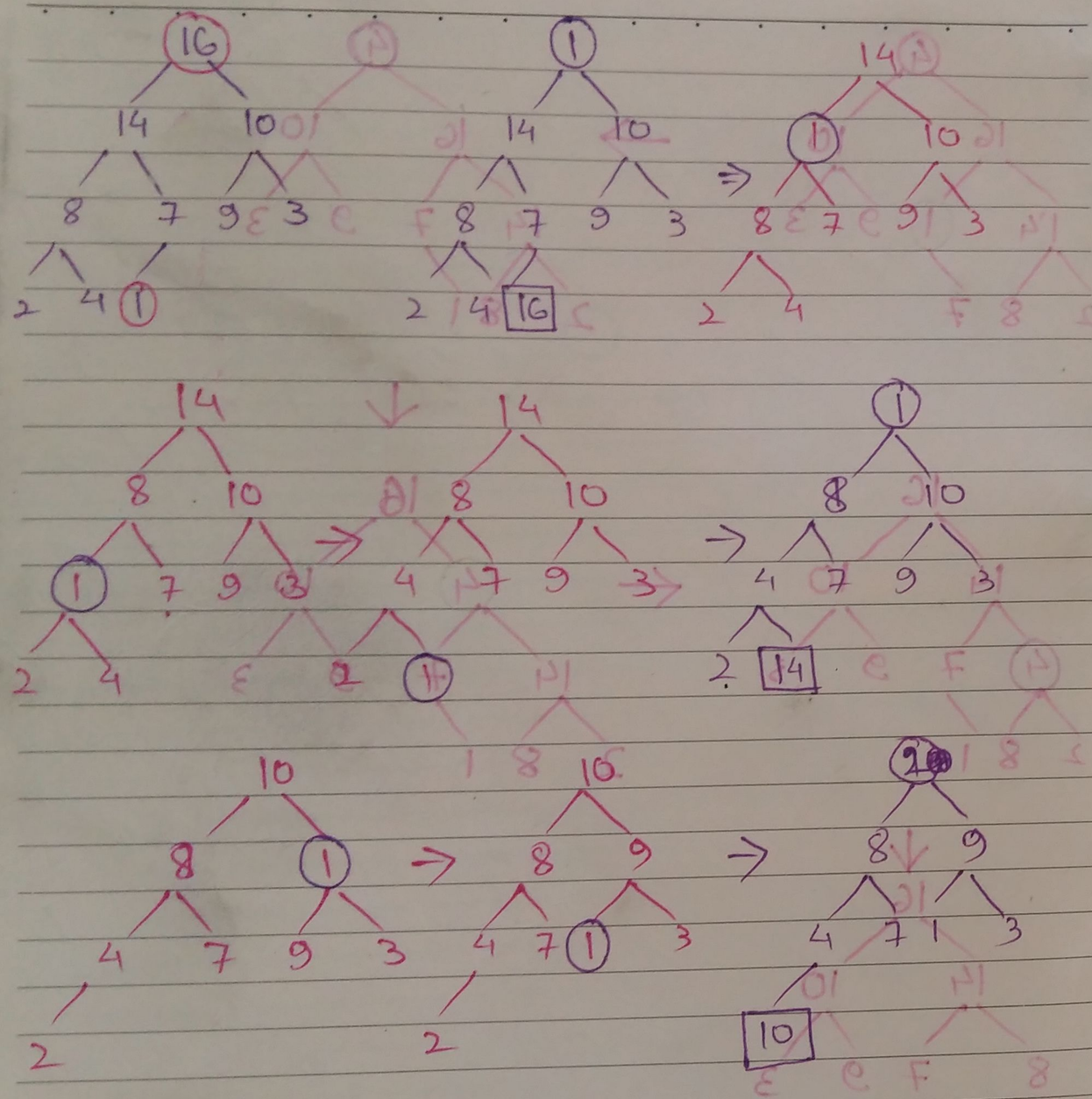


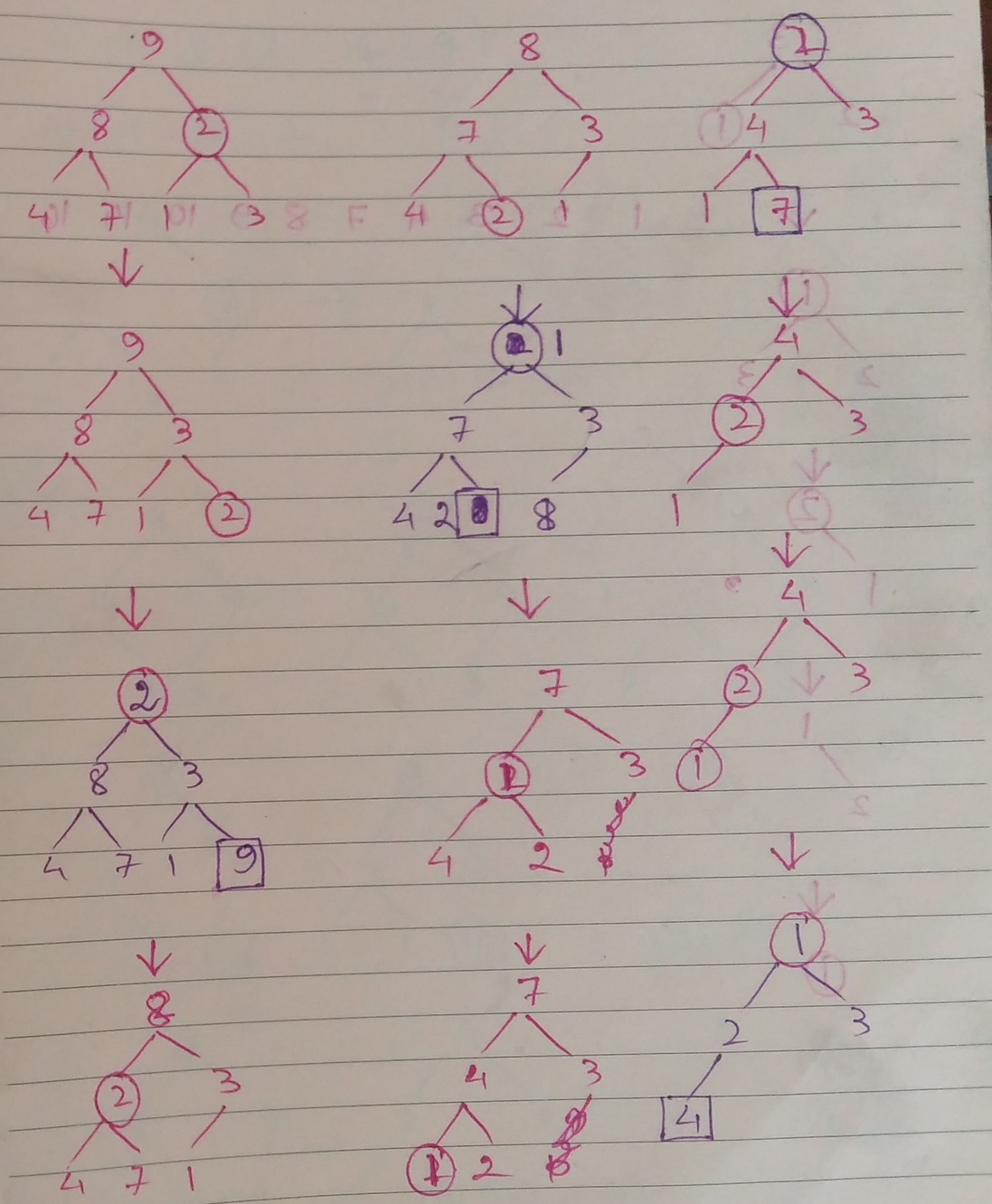


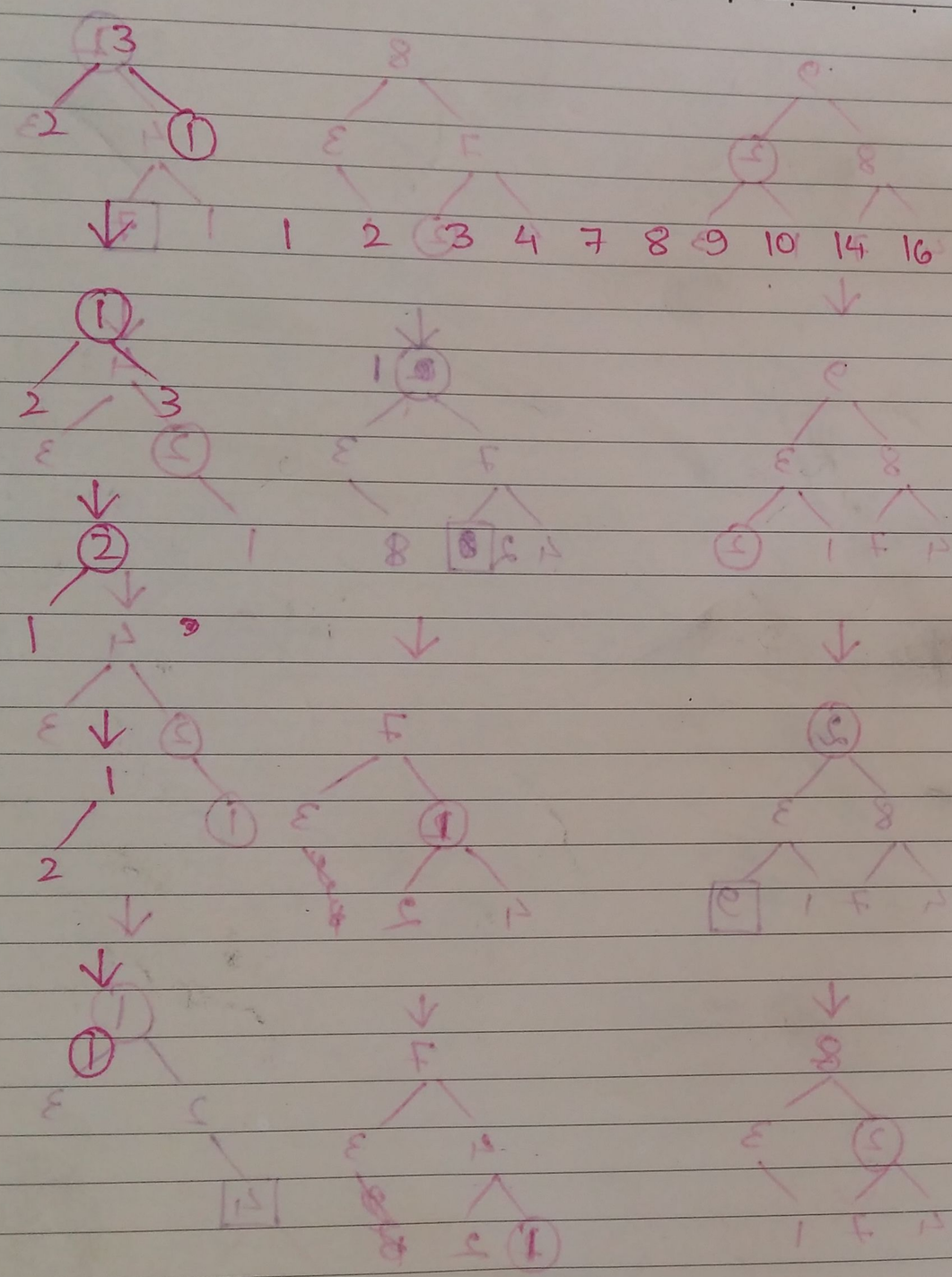
BUILD HEAP:


4
1 3
2 16 9 10
14 8 7
4
1 3
2 16 9 10
14 8 7
4
1 3
14 16 9 10
2 8 7
4
1 3
14 16 9 10
2 8 7
4
1 10
14 16 9 3
2 8 7
4
1 10
14 16 9 3
2 8 7


4
16 10
14 1 9 3
2 8 7
4
16 10
14 7 9 3
2 8 1
16
14 10
4 7 9 3
2 8 1
16
4 10
14 7 9 3
2 8 1
16
14 10
8 7 9 3
2 4 1



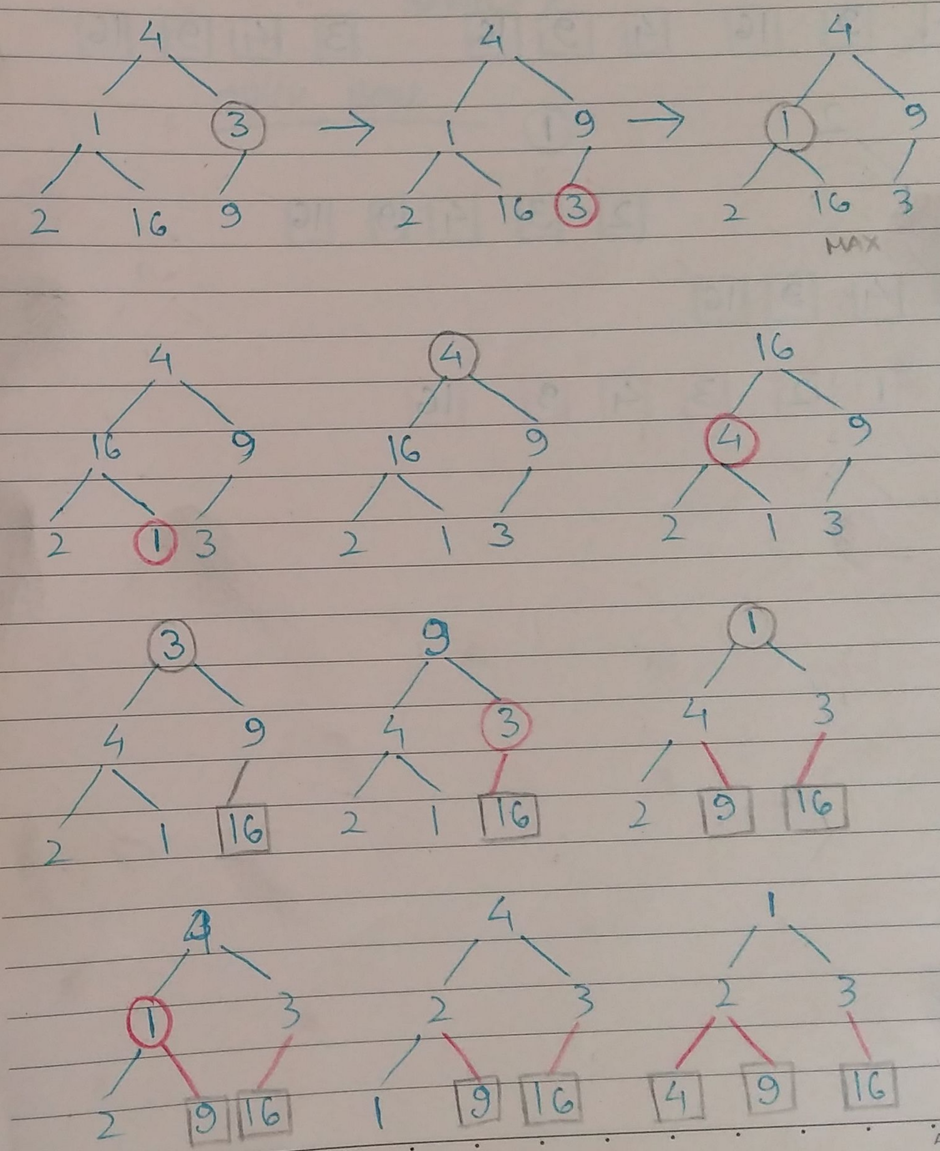


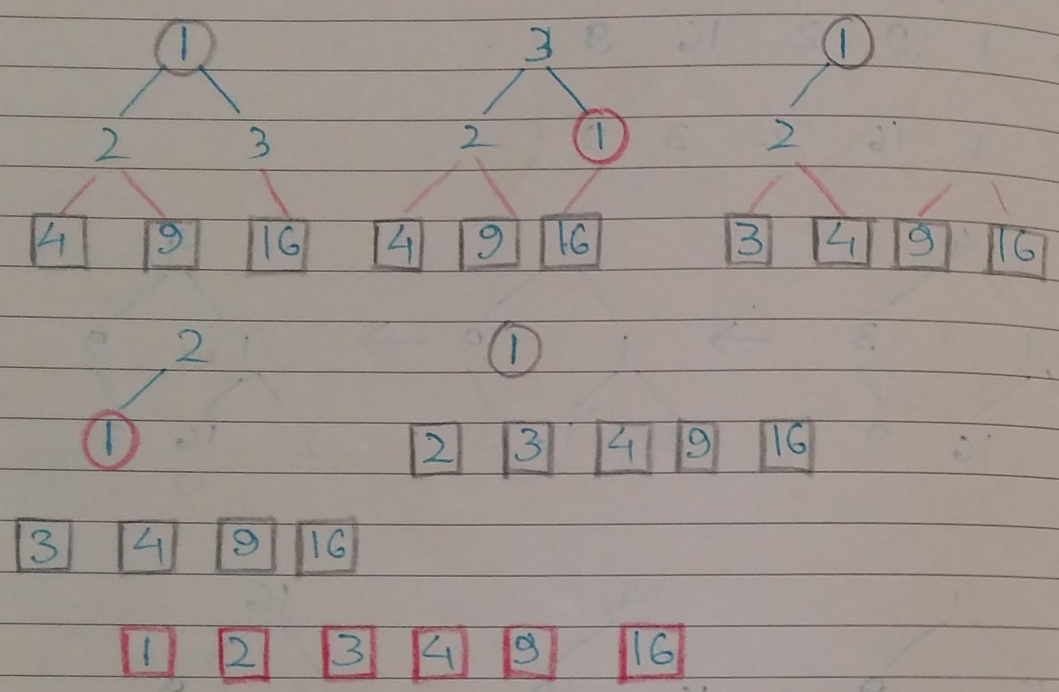


4 1 3 2 16 9

L R

4 1 16 2 3 9





SERCH → PARTS · STOCK

BANK TRANSCATON, PARTS/SERVICE / MOTOR CYCLE

QUICK BOOK

No.

Date

1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 15, 16, 17, 18, 19, 20, 3
21, 2, 3, 5, 6, 7, 30, 1, 4, 5, 6, 7, 9, 40, 24,
41, 42, 43, 45, 46, 48, 49, 50, 51, 53, 54,

11-42 08, 14,
11, 40

2, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19,
20, 2, 3, 5, 7, 31, 2, 4, 6, 8, 9, 40, 42, 3, 4, 6, 8, 9, 50
51, 53, 54
11-40, 14 08-14

22-JAN- 6 MST, SP,

18-JAN-
2, 3, 4, 5, 6, 7, 8, 9, 13, 15, 18, 21, 22, 23, 25, 26, 2
31, 32, 38, 39, 40, 42, 53, 54 11-520

19-JAN-
14, 25, 22, 2, 6, 4, 31, 11, 23, 34, 8, 32, 11-42,

A2M SUPREME

No.

Date