terms of time complexity in both cases.

→ when n is small, straight method.
n is large, Asymptotic method.

14/2/17.

DYNAMIC PROGRAMMING :

① MATRIX - CHAIN MULTIPLICATION :

matrix.	Dimensions
A	10 × 20
B	20 × 30
C	30 × 5

At a time we can multiply two matrices.

a) (AB)C ⇒ cost of multiplying = cost of $\overset{k}{(AB)}$ + cost of KC

= 10 × 20 × 30 + 10 × 30 × 5

= 6000 + 1500

= 7500

b) A(BC) = cost of $\overset{k}{(BC)}$ + cost of AK.

= 20 × 30 × 5 + 10 × 20 × 5

= 3000 + 1000

= 4000

Similarly if we want to multiply 4 matrices:

A	10×20
B	20×30
C	30×5
D	5×10

<u>Possibility</u>.

a) (((AB)C)D)

b) ((A(BC))D)

c) (A(BC)D)

d) (A(B(CD)))

e) ((AB)(CD))

Q) given chain of matrices $\{A_1, A_2 \dots A_n\}$ with A_i having dimension $P_{i-1} \times P_i$

dimension are stored in an array

$P = \langle P_0\ P_1\ P_2 \dots\dots P_n \rangle$

<u>Goal</u> To compute the product $A_1, A_2, A_3 \dots A_n$. as quickly possible (cost should be less).

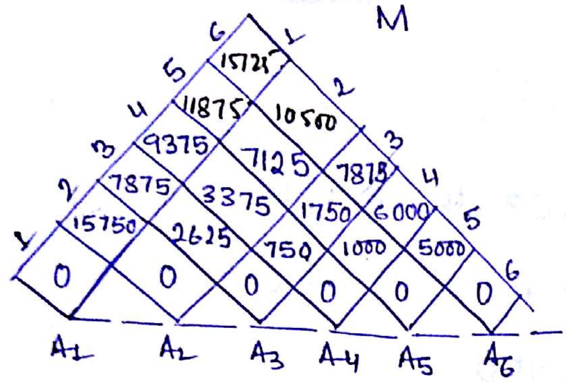
Find an optimal paranthesis of matrix chain product whose sequence of dimension is

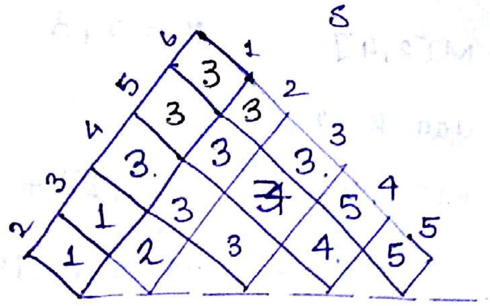
$$\langle \overset{P_0}{30}, \overset{P_1}{35}, \overset{P_2}{15}, \overset{P_3}{5}, \overset{P_4}{10}, \overset{P_5}{20}, \overset{P_6}{25} \rangle$$

↳ 7D : 6 matrices.

→ <u>Soln</u> <u>Step 1</u>.

construct a m table and a s table as follows.





Fill up the entries as per the formulae:

$$M[i][j] = \begin{cases} 0 & \text{if } i = j \\ \min\limits_{i \le k < j} \{ m[i][k] + m[k+1][j] + P_{i-1} P_k P_j \} & \text{if } i < j \end{cases}$$

$M[1,2]$ $k = 1$.

$M[1,2] = m[1,1] + m[2,2] + P_0 P_1 P_2$

$= 0 + 0 + 30 \times 35 \times 15$

$= 15750$

$M[2,3]$ $k = 2$.

$M[2,3] = 2625.$ $\{ m[2,2], m[3,3] + P_1 P_2 P_3 \}$

$M[3,4]$ $k = 3$.

$M[3,4] = 750$.

$M[1,3]$ $k = 1, 2$.

for $k = 1$.

$M[1,3] = \{ m[1,1] + m[2,3] + P_0 P_1 P_3 \}$.

$= \{ 2625 + 30 \times 35 \times 5 \}$

$= 7875$

for $k = 2$

$M[1,3] = 18000$

$M[2,4]$ $K = 2,3$

for $K = 2$.

$M[2,4] = \{ m[2,2] + m[3,4] + P_1 P_2 P_4 \}$

$= \{ \cancel{2625}\ 0 + 750 + 35.15.10 \}$

$= 5250 + \cancel{3375}\ 750$

$= \cancel{8625}\ 6000$

for $K = 3$

$M[2,4] = \{ m[2,3] + m[4,4] + P_2 P_3 P_4 \}$

$= \{ 2625 + 0 + 15.5.10 \}$

$= 3375$

$m[3,5]$. $K = 3,4$.

$M[3,5] = \{ M[3,3] + M[4,5] + P_2 P_3 P_5 \}$

for $K=3$ $= \{ 0 + 1000 + 15.5.20 \}$

$= 2500$

$M[3,5] = \{ M[3,4] + M[5,5] + P_3 P_4 P_5 \}$

for $K=4$ $= \{ 750 + 0 + 20.10.5 \}$.

$= 1750$

$M[4,6]$ $K = 4,5$

for $K=4$ $M[4,6] = \{ M[4,4] + M[5,6] + P_3 P_4 P_6 \}$

$= \{ 0 + 5000 + 5.10.25 \}$

$= 6250$

for $K=5$ $M[4,6] = \{ M[4,5] + M[6,6] + P_4 P_5 P_6 \}$

$= \{ 1000 + 0 + 25.20.10 \}$

$= 6000$.

$M[1,4] \quad K = 1,2,3.$

for $K = 1$. $\quad M[1,4] = \{ M[1,1] + M[2,4] + P_0 P_1 P_4 \}$.

$= \{ 0 + 3375 + 30.35.10 \}$.

$= \{ 13875 \}$.

for $K = 2$ $\quad M[1,4] = \{ M[1,2] + M[3,4] + P_0 P_2 P_4 \}$.

$= \{ 15750 + 750 + 30.15.10 \}$.

for $K = 3$ $\quad M[1,4] = \{ M[1,3] + M[4,4] + P_0 P_3 P_4 \}$

$= \{ 7875 + 0 + 30.5.10 \}$

$= 9375$

$M[2,5] \quad K = 2,3,4.$

for $K = 4$ $\quad M[2,5] = \{ M[2,4] + M[5,5] + P_1 P_4 P_5 \}$

$= \{ 9375 + 0 + 20.10.35 \}$

$= 10375$

$K = 3$ $\quad M[2,5] \longrightarrow \{ M[2,3] + M[4,5] + P_1 P_3 P_5 \}$.

$= \{ 7625 + 1000 + 35.5.20 \}$

$= \cancel{12375} \; 7125$

$K = 2.$ $\quad M[2,5] \longrightarrow \{ M[2,2] + M[3,5] + P_1 P_2 P_5 \}$

$= \{ 0 + 1750 + 35.15.20 \}$.

$= 1750 + 10500$

$= 12250$.

$M[3,6] \quad K = 3,4,5.$

for $K = 3$ $\quad M[3,6] = \{ M[3,3] + M[4,6] + P_2 P_3 P_6 \}$

$= 0 + 6000 + 25.15.5$

$= 7875$

for $K=4$ $\quad M[3,6] = M[3,4] + M[5,6] + P_2 P_4 P_6$

$= 750 + 5000 + 15.10.25$

$= 9500$

$K=5$ $\quad M[3,6] = M[3,5] + M[6,6] + P_2 P_5 P_6$

$= 1750 + 0 + 15.20.25$

$= 9250.$

$M[1,5]$ $\quad K = 1,2,3,4.$

for $K=1$ $\quad M[1,5] = \{ M[1,1] + M[2,5] + P_0 P_1 P_5 \}.$

$= \{ 0 + 7125 + 30.35.20 \}$

$= \{ 28125 \}.$

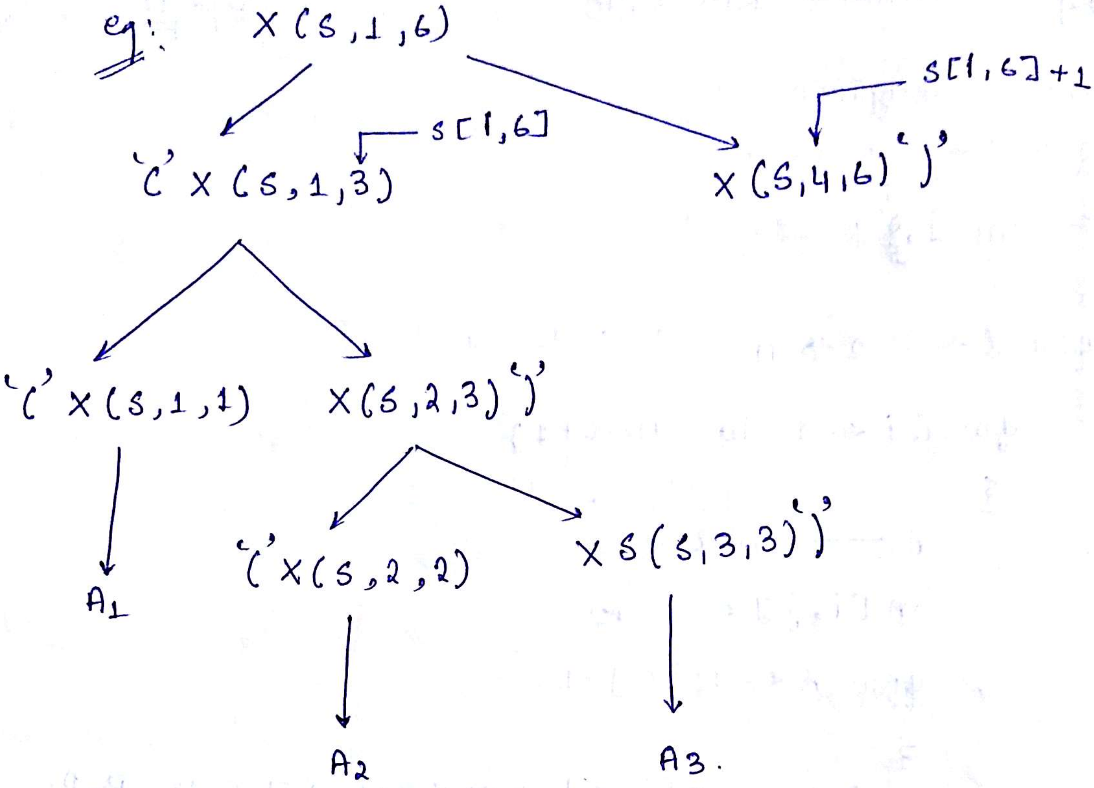
MA

{

Scanned by CamScanner

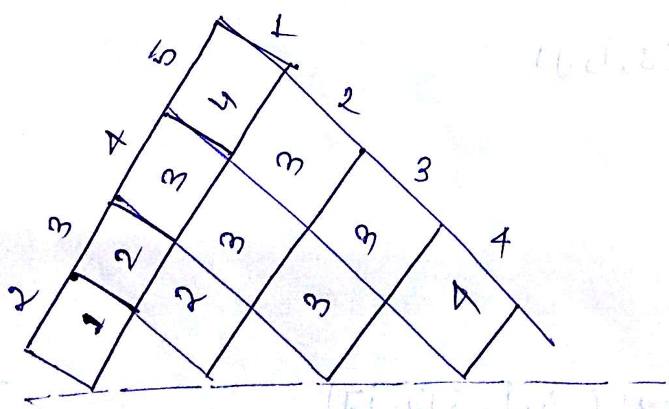
```
MATRIX - CHAIN - ORDER (P)                              15/2/17.
{
    n ← length [P] - 1
    for i ← 1 to n
    {    m[i, i] ← 0
    }
    for l ← 2 to n
    {
        for (i ← 1 to n-l+1)
        {
            j ← i + l - 1
            m[i, j] ← ∞
            for k ← 1 to j-1.
            {
                q ← m[i, k] + m[k+1, j] + P_{i-1} P_k P_j
                if q < m[i, j]
                {  m[i, j] ← q
                   s[i, j] ← k.
                }
            }
        }
    }
    return m & s ;

PRINT_OPTIMAL_PAREN (s, i, j)
{   if i == j
    Print "A";
    else
    {
        Print "(";
        PRINT_OPTIMAL_PAREN (s, i, s[i, j])
        PRINT_OPTIMAL_PAREN (s, s[i, j]+1, j)
        Print ')'
    }
}
```

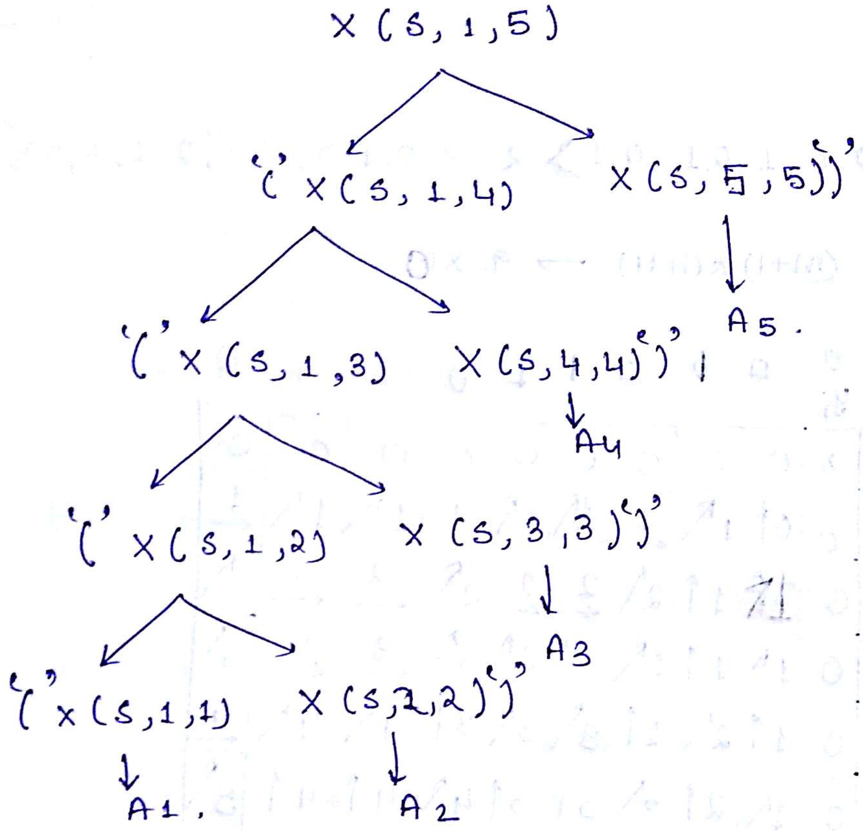
eg:



$((A_1(A_2A_3))((A_4\ A_5)\ A_6))$

Q:> Find the optimal parenthesization of a matrician product whose S table is given.





* $((((A_1 A_2) A_3) A_4) A_5)$.

17/2/17. DP

1

Find LCS. < 1, 0, 0, 1, 0, 1, 0, 1 > & < 0, 1, 0, 1, 1, 0, 1, 1, 0 >

$(m+1) \times (n+1) \longrightarrow 9 \times 10$

	y_i: 0	□	1	□	1	1	0	1	1	0
x_i	0	0	0	0	0	0	0	0	0	0
1	0	0 ↑	1 ↖	1 ←	1 ↖	1 ↖	1 ←	1 ↖	1 ↖	1 ←
0	0	1 ↖	1 ↑	2 ↖	2 ←	2 ←	2 ↖	2 ←	2 ←	2 ↖
0	0	1 ↖	1 ↑	2 ↖	2 ↑	2 ↑	3 ↖	3 ←	3 ←	3 ↖
1	0	1 ↑	2 ↖	2 ↑	3 ↖	3 ↖	3 ↑	4 ↖	4 ↖	4 ←
0	0	1 ↖	2 ↑	3 ↖	3 ↑	3 ↑	4 ↖	4 ↑	4 ↑	5 ↖
1	0	1 ↑	2 ↖	3 ↑	4 ↖	4 ↖	4 ↑	5 ↖	5 ↖	5 ↑

0

17

22/02/17

```
LCS_LENGTH (X,Y)
{
    m <—— length [X]
    n <—— Length [Y]
    for i <—— 1 to m
    c[i,0] <—— 0
    for j <—— 1 to n
    c [0, j] <—— 0
    for i <—— 1 to m
    {
        for j <—— 1 to n
        {
            if (xi == yi)
            {
                c[i, j] = c[i-1, j-1]+1
                b[i, j] = "↖";
            }
            else if  c[i-1, j] >= c[i, j-1]
                {
                c[i, j] = c[i-1, j]
                b[i, j] = "↑";
                }
                else
                {
                c [i, j] = c[i, j-1]
                b [i, j] = "←";
                }
        }
    }
    return c and b
}.
```

```
PRINT_LCS (b, x, i, j)
{
    if i == 0 or j == 0
    then return
    if b[i, j] = "↖"
    {
        PRINT_LCS(b, x, i-1, j-1);
        Print "x" i
    }
    else if b[i, j] = "↑"
            PRINT_LCS (b, x, i-1, j);
        else
            PRINT_LCS (b, x, i, j-1);
}.
```

28/02/17

GREEDY ALGORITHMS

① KNAPSHACK PROBLEMS :

→ A thief robbing a store finds n items in a store. The quantity (kg) and market price/value ~~coupes~~ of available items are ($w_1, w_2, \ldots w_n$) and ($v_1, v_2 \ldots v_n$) respectively. (max. load W)

→ Two strategies used are :

0/1 knapshack.	fractional knapshack.
i) Here the items cannot be broken	Here the items may be broken.
ii) ~~solved by dynamic~~ programming strategy.	can be solved by greedy approach.

→ Problem : method applied so as to carry maximum load by thief.

Ex	30 Kg Gold.	~~100~~ 40 kg Iron	50 kg Silver.	W = 85 Kg.
Price →	60000	70000	50000	

Dynamic approach.

x_1	x_2	x_3	
1	1	0	→ 67000
1	0	1	→ 110000

x_1	x_2	x_3	
1	1	15/50	→ 82000
0	35/40	1	→ 56125
1	5/40	1	→ 110875

Scanned by CamScanner

eg) Find an optimal soln to the knapsack instance $n=7$, $W=15$

$(v_1, v_2, \ldots v_7) = (10, 5, 15, 7, 6, 18, 3)$ and

$(w_1, w_2, \ldots w_7) = (2, 3, 5, 7, 1, 4, 1)$

soln.

Item	v_i	w_i	v_i/w_i
1.	10	2	5
2.	5	3	1.66
3.	15	5	~~3~~
4	7	7	1
5.	6	1	6
6	18 7	4	4.5
7	3	1	3.

sort descending as per v_i/w_i, $U = W = 15$

Item	v_i	w_i	v_i/w_i	Remark	x_i
5	6	1	6	$w_5 \le U$ $U = 15-1$	1.
1	10	2	5	$w_1 < U$ $U = 14-2$	1
6	8	4	4.5	$w_6 < U$ $U = 12-4$	1
7	3	1	3	$w_7 < U$ $U = 8-1$	1
3	15	5	3	$w_3 < U$ $U = 7-5$	1
2	5	3	1.66	$w_2 > U$ $U = 0$	2/3
4	7	7	1		0

soln Vector

$\langle 1, 2/3, 1, 0, 1, 1, 1 \rangle$.

1/03/17

GREEDY_KNAPSACK (W, n)
{
/* V[1..n] and W[1..n] are two arrays representing profit or value and weight of n items respectively. W is the weight of knapsack. x[1..n] is the soln vector. Items are arranged in descending order of their values. */

for i ← 1 to n
x[i] ← 0
u ← W
for i ← 1 to n
{ if (wi ≤ u)
x[i] ← 1
u = u - wi
}
else
break;
}
if (i ≤ n)
x[i] ← u/wi
}.

Q) Find an optimal soln for the following knapsack instance.

n = 8   W = 17   <Profit> P = <10, 15, 8, 7, 3, 15, 8, 27>

W = <5, 4, 3, 7, 2, 3, 2, 6>

item	Vi	Wi	Vi/Wi
1	10	5	2
2	15	4	3.75
3	8	3	2.67
4	7	7	1
5	3	2	1.5
6	15	3	5
7	8	2	4
8	27	6	4.5

item	Vi	Wi	Vi/Wi	R	Xi
6	15	3	5	17-3	3
8	27	6	4.5	14-6	6
7	8	2	4	8-2	2
2	15	4	3.75	6-4	4
3	8	3	2.67	2/3	2/3
1	10	5	2		0
5	3	2	1.5		0
4	7	7	1		0

$S = \langle 0, \cancel{1}, 2/3, 0, 0, \cancel{1}, \cancel{1}, \cancel{1} \rangle$

HA: If the name of items are $A_1, A_2, \ldots, A_n$. the WAP by using structure to solve the greedy knapsack problem. Display the ~~Set~~ result as soln vector and also in tabular format

2/03/17

	1	2	3	4
PH ⇒	D	G	B	F

feasible soln.

	A	B	C	D	E	F	G
P_i	10	50	30	40	25	35	45
di	1	3	2	3	4	4	2

A B D E = 10+50+40+25
A B D F
D G B F
D G B E

B A C D
A B C G

Not feasible.

JOB Sequencing with deadline:

- There are n jobs to be processed in a processor each at a time. Each job is associated with a deadline $d \geq 0$ and $P > 0$ where P & d are integers.
- The problem is to arrange/schedule these n jobs in a single machine with the constraint is called job sequencing deadlines.

$P = \{20, 15, 10, 5, 1\}$

$d = \{2, 2, 1, 3, 3\}$.

2	1	3

	P_i	d_i	Select
1	20	2	✓
2	15	2	✓
3	10	1	
4	5	3	✓
5	1	3	

→ Schedule the following jobs to get max profit. Each job takes 2 hrs.

Job no :	1	2	3	4	5	6	7	8
Profit :	15	18	4	25	3	4	10	6
deadline :	7	5	6	2	4	2	3	4.

Soln

0 - 2	2 - 4	4 - 6	6 - 8
4	2	1.	

JNo	P_i	d_i	S
4	25	2	✓
2	18	5	✓
1	15	7	✓
7	10	3	
8	6	4	
6	4	2	
3	4	6	
5	3	4.	

Opt Order : 4, 2, 1.

Profit : $25 + 18 + 15 = 58$

07/03/17

ACTIVITY SELECTION PROBLEM:

$S = \{a_1, a_2 \dots a_n\}$ of n proposed activities that wish to use a resourse that can be used by one activity at a time. Each activity a_i is defined by a pair consisting of a start time s_i and finishing time f_i where $0 \leq s_i \leq f_i < \infty$

Activity selection problem is to select a maxm size subset of mutually compatible activities. (non conflicting activities).

eg:
Given 11 activities along with their time span as given as follows.

i	1	2	3	4	5	6	7	8	9	10	11
a_i	k	j	i	h	g	f	e	d	c	b	a
s_i	12	2	8	8	6	5	3	5	0	3	1.
f_i	14	13	12	11	10	9	8	7	6	5	4.

Soln. Arrange activities in ascending order as per finishing time of each activities.

i	ai	si	fi	Selection.
1	a	1	4	✓
2	b	3	5	×
3	c	0	6	×
4	d	5	7	✓
5	e	3	8	×
6	f	5	9	×
7	g	6	10	×
8	h	8	11	✓
9	i	8	12	×
10	j	2	13	×
11	k	12	14	✓

$\langle a, d, h, k \rangle$

Q.)

i	1	2	3	4	5	6	7	8	9	10	11	12
ai	a	b	c	d	e	f	g	h	i	j	k	l
Si	44	7	37	83	27	49	16	44	44	58	27	26
fi	86	25	96	89	84	62	17	~~88~~ 70	84	94	79	57.

i	ai	si	fi	Selection.
1	g	16	17	✓
2	b	7	25	
3	l	26	57	✓
4	f	49	62	
5	h	~~23~~ 44	70	
6	k	27	79	
7	e	27	84	
8	i	44	84	
9	a	44	86	
10	d	83	89	✓
11	j	58	94	
12	c	37	96.	

Soln

A = {g, l, d}

```
GREEDY_ACTIVITY_SELECTOR (s, f).
{
      /* s[1...n] and f[1...n] contains start and finish
         time resp. A subset A is the soln vector
         contains max, non conflicting activities */

      n <------ length [s];
      i <---- 1;
      A = {a1};
      for m <---- 2 to n
      {    if Sm >= fi
           {
              A <---- A U {Am};
              i <---- m;
           }
      }
      return A.
}
```

15/03/17

HUFFMAN CODE

coding is the problem of representing data in another form. (generally encoded in binary).

encoding of a character is called codeword.

Different types of coding :

- Fixed length code : Each codeword uses same no. of bits.
- Variable length code : Each codeword can use diff. no. of bits.
- Prefix code : No code word is a prefix of any other codeword
- Huffman code: Huffman invented an greedy algo. that can construct an optimal prefix code.

Q.) Variable length code can do better than fixed length code. Justify!!

→ size of records will be small.
records will need small storage space.
records load faster.

] Variable length code adv.

16/03/17

Huffman Algorithm.

```
HUFFMAN (C)
{
  /* C is a set of characters where each c ∈ C
  is an object of defined frequency f(c). Q is a
  minimum priority queue */

  n ← |C|
  Q ← C          // create a min heap with set of freq.

  for i=1 to n-1
  {   do allocate a new node z
      left [z] ← x ← EXTRACT_MIN (Q)
      right [z] ← y ← EXTRACT_MIN (Q)
      f[z] = f[x] + f[y].
      INSERT (Q, z)
  }
  return EXTRACT_MIN (Q)   // Returns root of tree.
}
```

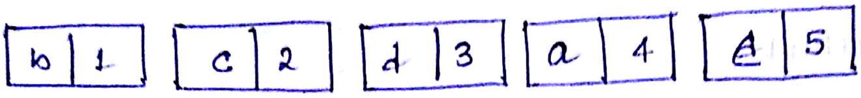
Q.> const. a Hoffman code for the following data.

char	a	b	c	d	e
freq	4	1	2	3	5

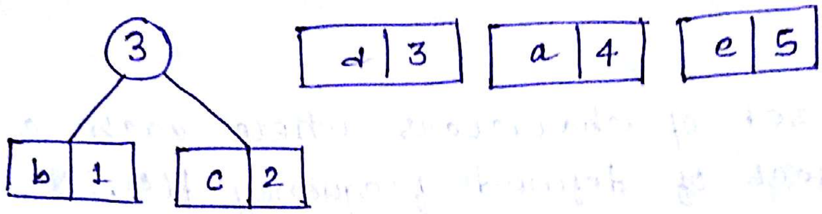
How many bricks are needed to encode a string containing 12 a's, 7b's, 5c's, 3d's & 4e's using this code compare with another code where each character is encoded with fixed no. of bits.

Sol^n Part 1.

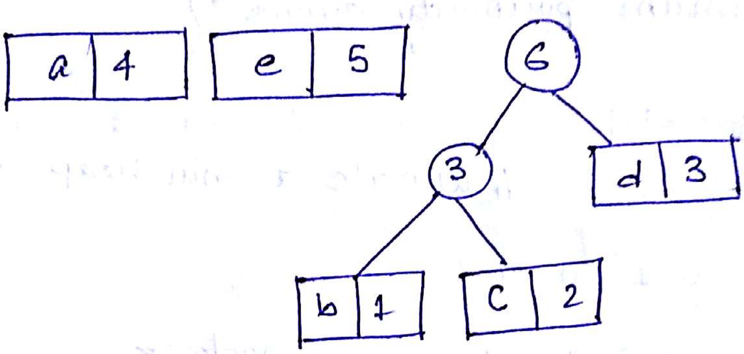
~~Start~~ #



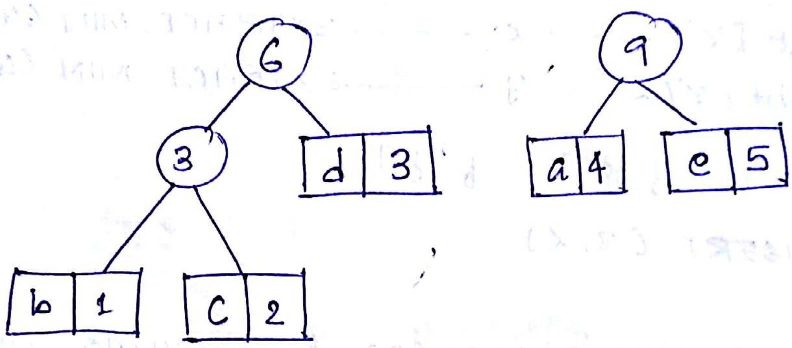
S1 #



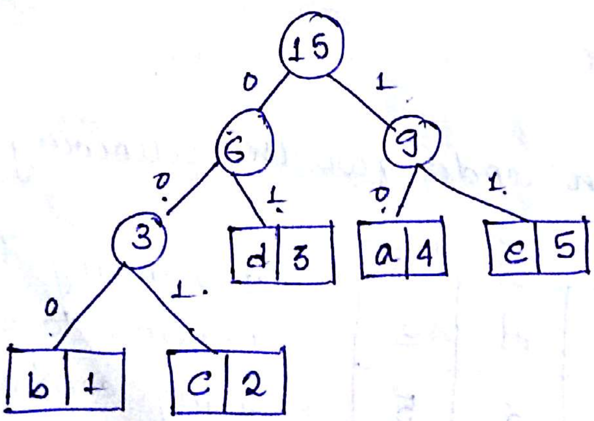
S2 #



S3 #



S4 #



Huffman code :

a — 10

b — 000

c — 001

d — 01

e — 11

How to code b a d a e.
↓ ↓ ↓ ↓ ↓ : write their corresponding code so found.
000 10 01 10 11

How to decode <u>001</u> <u>000</u> <u>10</u> <u>000</u> <u>01</u> <u>11</u> <u>01</u>. Start comparing from left with code so found.
c b a b d e a

Bits req → 11 (Huffman)
in fixed code ∴ 5/2 ⇒ 3 per char ∴ 15 bits.

Sol^n Part 2 HC : 12 × 2 + 7 × 3 + 5 × 3 + 3 × 2 + 4 × 2
→ 24 + 21 + 15 + 6 + 8
→ 74.

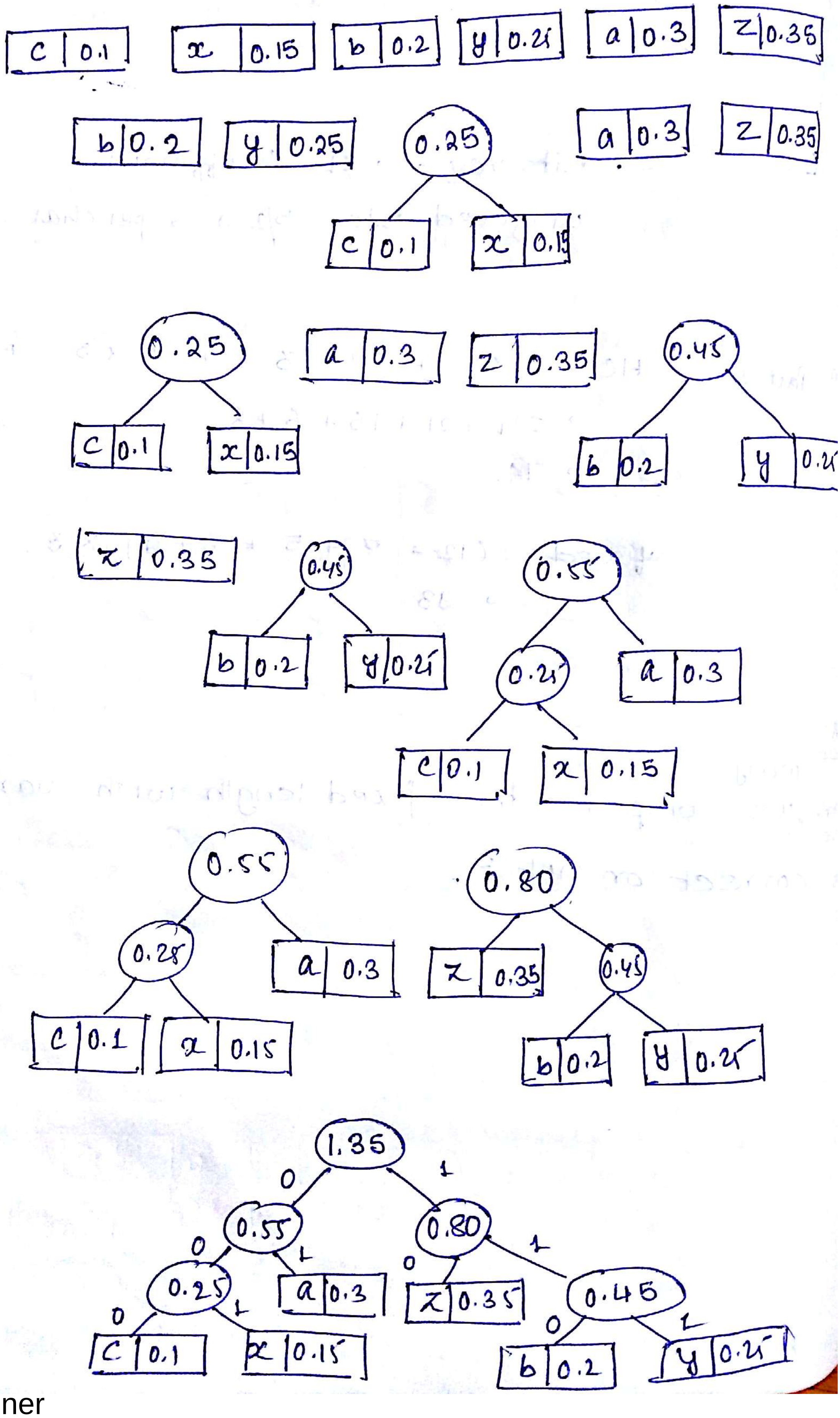
fixed : (12 + 7 + 5 + 3 + 4) × 3.
→ 93.

HA.
~~Can~~ The way we compare the fixed length with variable length. is correct or not ?

Scanned by CamScanner

Ex 2

char	a	b	c	x	y	z
Freq	0.3	0.2	0.1	0.15	0.25	0.35



a — 01
b — 110
c — 000
x — 001
y — 111
z — 10

~~Given~~ 10 a's 15 b's 7c's ~~codes~~
10 x's 2 y's ~~codes~~
5 z's

$10 \times 2 + 15 \times 3 + 7 \times 3$
$+ 10 \times 3 + 2 \times 3 + 5 \times 2$
$\longrightarrow 20 + 45 + 21 + 30 + 6 + 10$
$\longrightarrow 132$

~~Code~~ .

Fixed length : $(10 + 15 + 7 + 10 + 2 + 5) \times 3$
$\longrightarrow 49 \times 3$
$\longrightarrow 147$

20|3|17

OPTIMAL STORAGE ON TAPE PROBLEM :

Let 2 programs have lengths / runtime of 6 & 3.
$P_1 = 6$ & $P_2 = 3$.

Possible orders.

$P_1 P_2$ Retrieval time $\underset{P_1}{6} \mid \underset{P_2}{3} \rightarrow$ RT = 6 + (6 + 3) = 15.
$P_2 P_1$ Retrieval time $\underset{P_2}{3} \mid \underset{P_1}{6} \rightarrow$ RT = 3 + (3 + 6) = 12

∴ P_2 accessed only after P_1 is executed.

eg:

P_1	P_2	P_3
6	3	4

Find the min retrieval time of comb.

soln

$P_1 P_2 P_3$ 6 + (6 + 3) + (6 + 3 + 4) = 28
$P_2 P_3 P_1$ 3 + (3 + 4) + (3 + 4 + 6) = 23 —— optimal soln
$P_3 P_1 P_2$ 4 + (4 + 6) + (3 + 4 + 6) = 27
$P_1 P_3 P_2$ 6 + (6 + 4) + (6 + 4 + 3) = 29
$P_2 P_1 P_3$ 3 + (3 + 6) + (3 + 6 + 4) = 25
$P_3 P_2 P_1$ 4 + (4 + 3) + (4 + 3 + 6) = 24

Q) Find optimal order of n programs having lengths
10, 5, 7, 11, 2, 3, 8, 9, 1, 8

Sol^n Step 1 : arrange in ascending order of length.

1, 2, 3, 5, 7, 8, 8, 9, 10, 11 .

mean retrieval time = $\frac{1}{10}$ (1 + (1+2) + (1+2+3) +

Many tape devices (M) & many programs (n)

(All tapes are read concurrently)

2 tape drives

2 programs of length 6 & 3

Approaches

1. T1 [P1 | P2] 12   T2 [|] 0 ⟶ 12
2. T1 [|] 0   T2 [P2 | P1] 12 ⟶ 12
3. T1 [P2 |] 3   T2 [P1 |] 6 ⟶ 9
4. T1 [P1 |] 6   T2 [P2 |] 3 ⟶ 9

eg: T ⟶ 2   P ⟶ 3

P_1	P_2	P_3
6	3	2

Sol^n

1. T_1 [P3 | P2 |] 7 T_2 [P1 | |] 6   ] — 13

2. T_1 [P2 | P1 |] 12 T_2 [P3 | |] 2   ] — 14

3. T_1 [P3 | P1 |] 10 T_2 [P2 | |] 3   ] — 13.

eg: T ⟶ 2   P ⟶ 4

P_1	P_2	P_3	P_4
2	3	5	7

a)

T_1	P_1	P_2	P_3		17
T_2	P_4				7

→ 24

b)

T_1	P_1	P_2			7
T_2	P_3	P_4			17

→ 24

c)

T_1	P_1	P_3			9
T_2	P_2	P_4			13

→ 22

d)

T_1	P_1	P_4			11
T_2	P_2	P_3			11

→ 22

e)

T_1	P_1	P_3	P_4		23
T_2	P_2				3

→ 26

Arrange in ascending order
store odd no. programs in one table and even in other

Q) Find optimal storage for 13 programs on 3 magnetic tapes.

12, 5, 8, 32, 7, 5, 18, 26, 4, 3, 11, 10, 6

⟶ 3, 4, 5, 5, 6, 7, 8, 10, 11, 12, 18, 26, 32

T1	3	5	8	12	32
T2	4	6	10	18	
T3	5	7	11	26	

21/3/17.

SHORTEST PATH PROBLEM (SPP)

- To find shortest path between any two nodes or vertices of a given weighted group.
- There are many variations in SPP

SPP

Single source shortest path (SSSP) problem or single source All Destination (SSAD)
- ~~Dijkstra's Algorithm~~
- Bellman Ford Algorithm. (Dynamic)

All pair shortest path (APSP) problem
- can be solved by Floyd-Warshall's Algo (Dynamic)

Initializing Single source shortest Path problem.

```
INITIALIZE - SINGLE SOURCE (G, s)
{   for all v ∈ V do
    {   d[v] = ∞
        π[v] = 0
    }
    d[s] = 0
}
```

d[v] = shortest path found so far given source vertex s ~~from~~ to V

Relaxing an edge

- Testing whether we can get any improvement of d[v], if so update d[v] else ignore.

```
RELAX (u, v, w)
{
   if d[u] + w(u,v) < d
   {
      d[v] = d[u] + 
      π[v] = u
   }
}
```

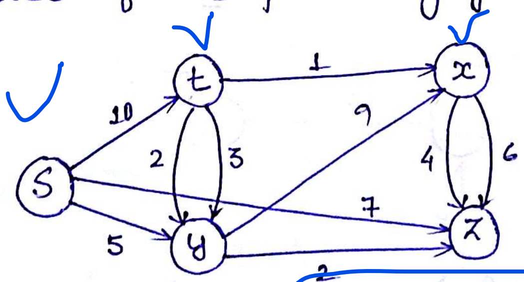
$\pi[v]$ = predecessor vertex of V

DIJKSTRA's ALGO:-

Imp

```
DIJKSTRA (G, W, S)
{
    INITIALIZE_SINGLESOURCE (G,S)
    S <-- φ
    Q <-- V[G]    // const a min-heap on Q
    while Q ≠ φ
    {
        u <-- EXTRA_MIN (Q)
        S <-- S ∪ {u}        union
        for each vertex V ∈ Adj [U]
        RELAX (U,V,W)
    }
}
```

Q.) Use the above algorithm to find shortest path from s to other vertices of the following graph.



Sol^n

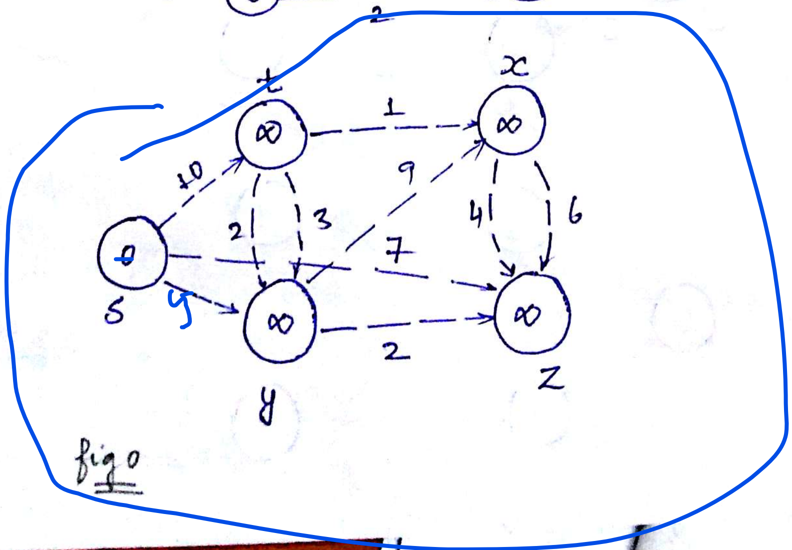


fig 0

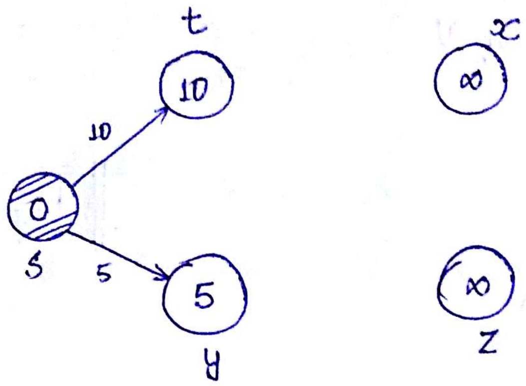


fig 1

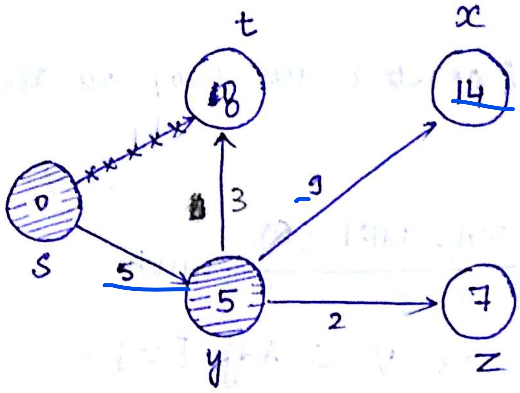


fig 2

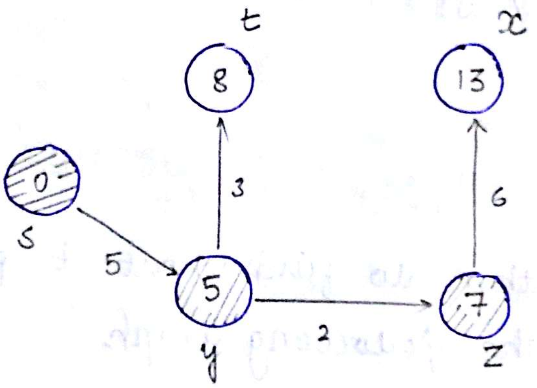


fig 3

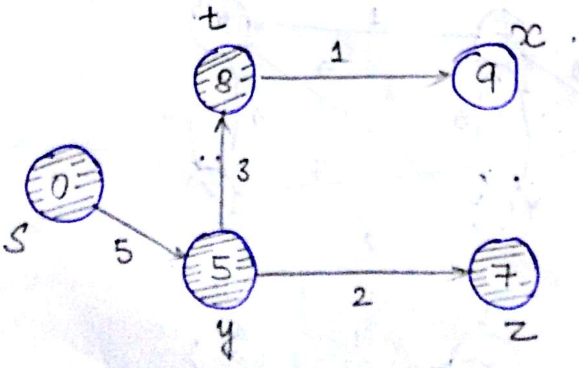


fig 4

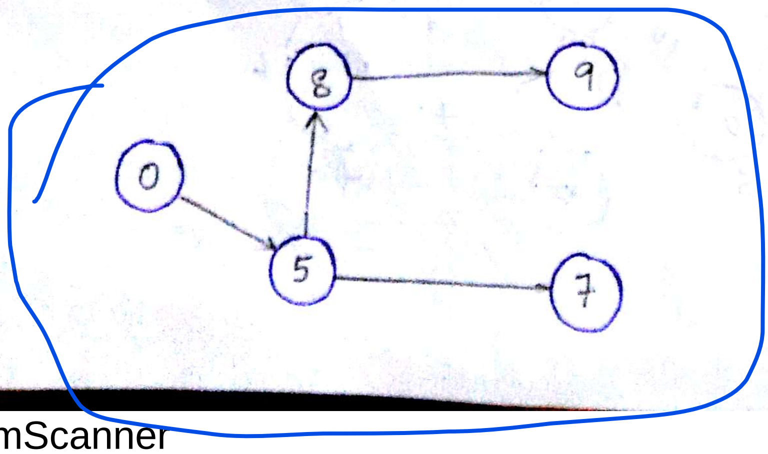


fig 5

minimu

s t
s t
s t
s

HA

use su

A ito C

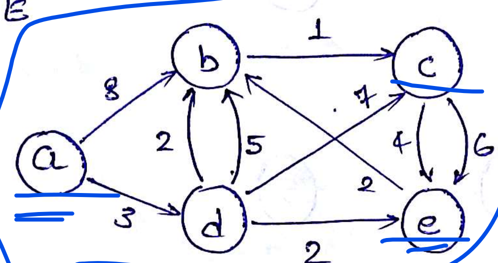
minimum distance.

s to t ⟶ 8
s to y ⟶ 5
s to x ⟶ 9
s to z ⟶ 7.

HA

use suitable shortest path algo to find shortest path b/w

A to C to A to E



Sol^n

Fig 0

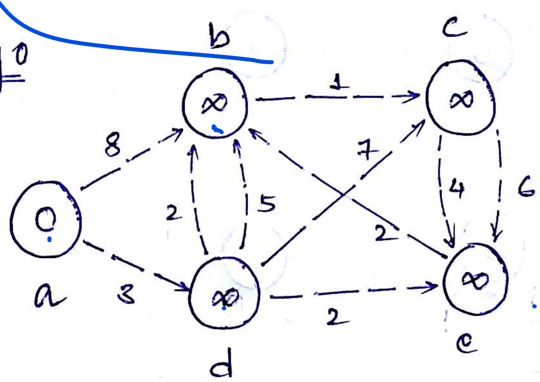


Fig 1.

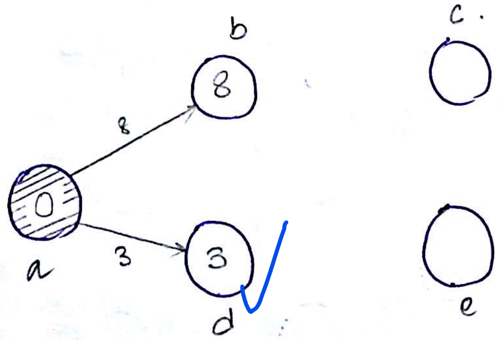


Fig 2

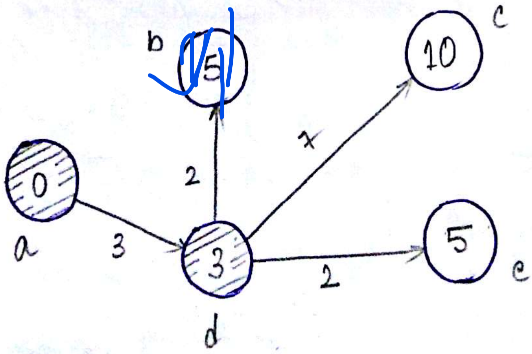
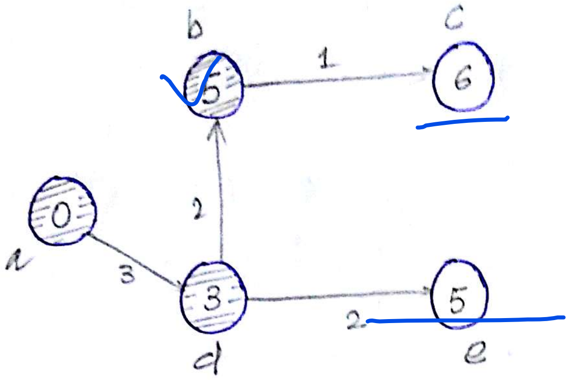
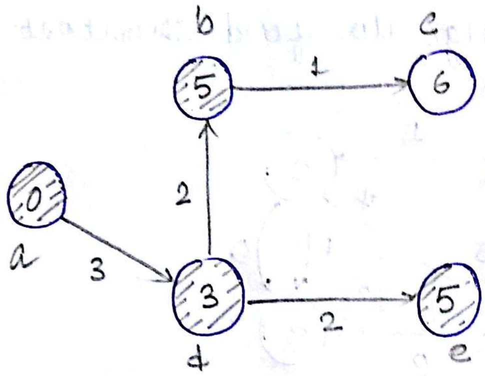


fig 3



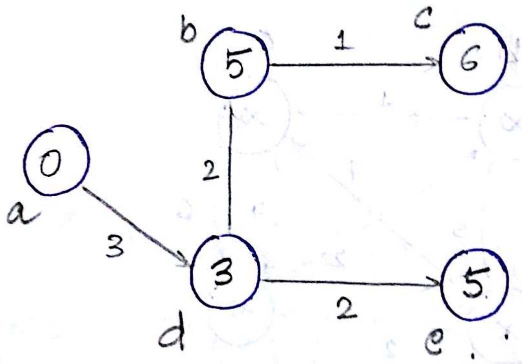
b → c ✓

fig 4



e → b ×

fig 5

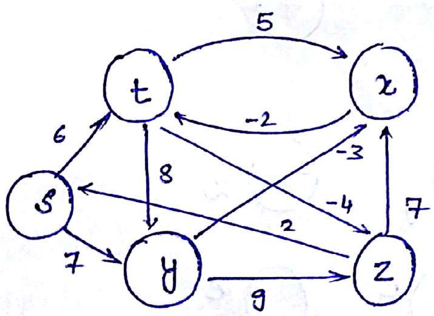


23/3/17

BELLMAN - FORD ALGO.

```
Bellman-ford (G, w, s)
{  // initialize step
   INITIALIZE-SINGLE-STEP (G, S)
   // relax step.
   for i <— 1 to |V[G]|-1
   {
        for eachedge(U,V) ∈ E[G]
        RELAX (U,V,w)
   }
   // Testing step for -ve weight cycle
    for eachedge (U,V)  ∈ E[G].
    {   if d[V] > d[U] + w(U,V)
        return false
    }
    return TRUE.
}
```

Ex 1. Run bellman ford algo on the given graph.



Solⁿ In each pairs relax the edges in order (t, x), (t, y), (t, z), (x, t), (y, x), (y, z) (z, x), (z, s), (s, t), (s, y) and π values after each pairs.

	f1	f2	f3	f4
(t,x)	×	✓	×	×
(t,y)	×	×	×	×
(t,z)	×	✓	×	✓
(x,t)	×	×	✓	×
(y,x)	×	✓	×	×
(y,z)	×	×	×	×
(z,s)	×	×	×	×
(s,t)	✓	×	×	×
⇐ (s,y)	✓	×	×	×

fig 0

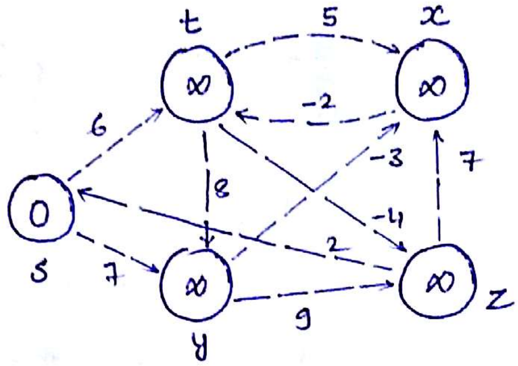


fig 1

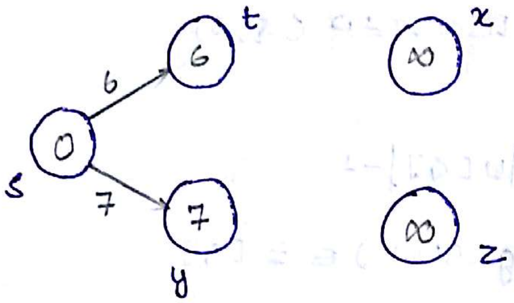


fig 2

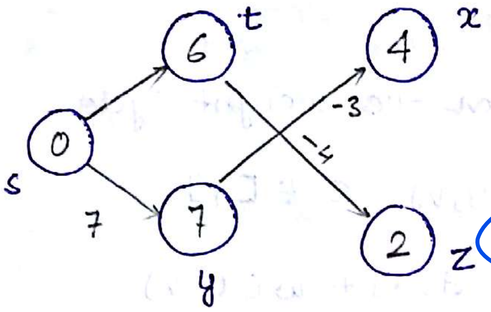


fig 3

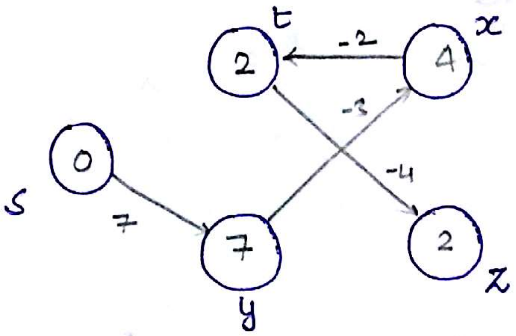


fig 4

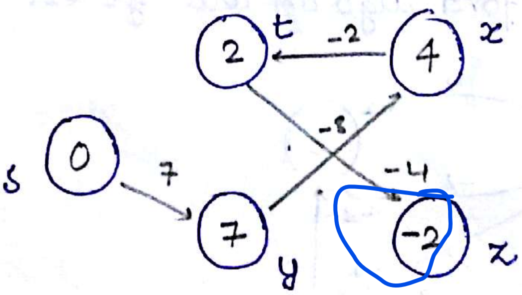
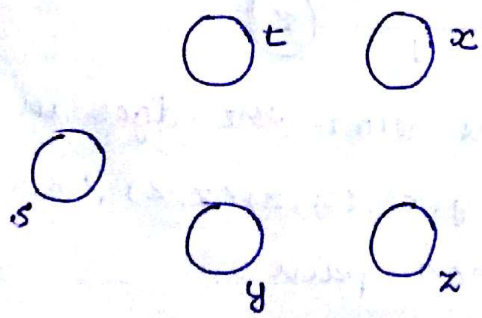


fig 5



Differentiate b/w −ve weight cycle & +ve weight cycle?

24/03/17

FLOYD - WARSHALL's ALGO

TO SOLVE APSP BY USING DP

FLOYD - WARSHAL (W)

{ $n \longleftarrow$ rows [W]

$D^{(0)} \longleftarrow W$

for $k \longleftarrow 1$ to n

{ for $i \longleftarrow 1$ to n {

for $j \longleftarrow 1$ to n

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

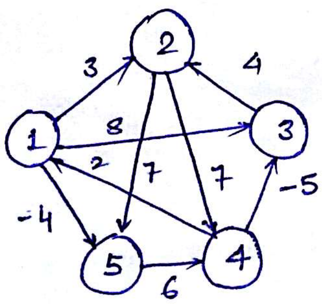
}}

}

return $D^{(n)}$

}

Example : Run the Floyd Warshall's algo on the following graph & show the shortest path



Soln

$D^{(0)}$	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	∞	-5	0	∞
5	∞	∞	∞	6	0

$D^{(1)}$	1	2	3	4	5
1	0	3	8	∞	-4
2	∞	0	∞	1	7
3	∞	4	0	∞	∞
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(2)}$	1	2	3	4	5
1	0	3	8	4	-4
2	∞	0	∞	1	7
3	∞	4	0	5	11
4	2	5	-5	0	-2
5	∞	∞	∞	6	0

$D^{(3)}$	1	2	3	4	5
1	0	3	8	4	-4
2		0			
3			0		
4				0	
5					0

$D^{(4)}$	1	2	3	4	5
1	0				
2		0			
3			0		
4				0	
5					0

$D^{(5)}$	1	2	3	4	5
1	0				
2		0			
3			0		
4				0	
5					0

Minimum Spanning Tree (MST)

28/03/17

Spanning Tree (ST) A subgraph of a graph that contains all vertices and some edges without having a cycle.

Weighted Spanning Tree (WST)

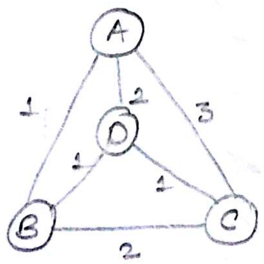
ST + edges having lables (weights)

cost of WST = Sum of all edges of WST.

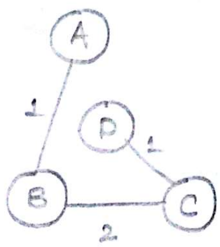
Minimum cost spanning tree or MST.

The min. cost of the spanning tree found among all possible spanning-trees is called a MST.

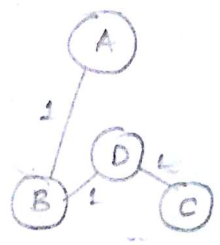
Ex.



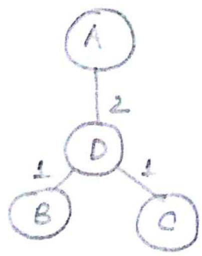
⇒



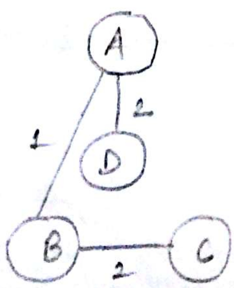
G1 : cost : 4



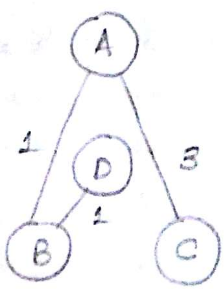
G2 : cost : 3



G3 : cost : 4



G4 : cost : 5



G5 : cost : 5

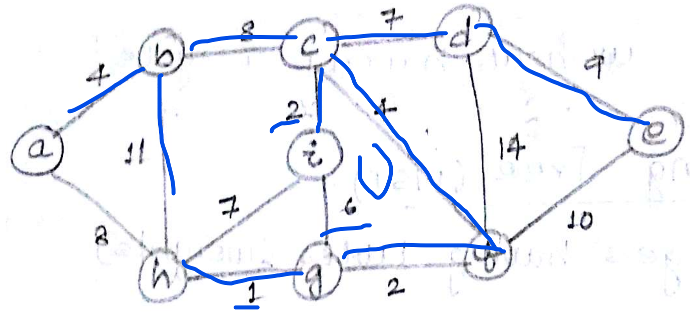
— Two Algo to solve MST

a) Krushal's Algo

b) Prim's Algo.

KRUSHAL's ALGO :

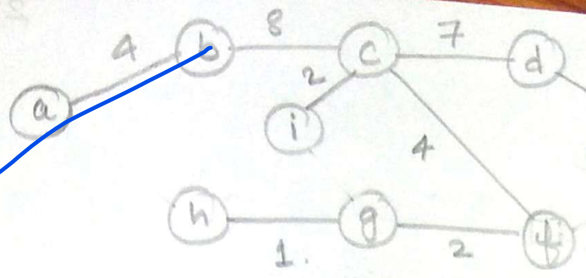
Ex Findout the MST by krushal's Algo.



sol.

Sort the edges as per their weights in increasing order.

Sl.no.	edge	weight	Selection
1.	(h,g)	1.	✓
2.	(g,f)	2	✓
3.	(i,c)	2	✓
4.	(a,b)	4	✓
5.	(c,f)	4	✓
6.	(i,g)	6	✗
7.	(h,i)	7	✗
8.	(c,d)	7	✓
9.	(b,c)	8	✓
10	(a,h)	8	✗
11.	(d,e)	9	✓
12.	(f,e)	10	✗
13.	(b,h)	11	✗
14.	(f,d)	14	✗



cost: 1 + 2 + 2 + 4 + 4 + 7 + 8 + 9

: 37

KRUSHKAL (G, w)

{

A ⟵ Φ

for each vertex $v \in V[G]$

{

MAKE_SET (v)

}

Sort the edges of E by non-decreasing wt. w.

for each ege (u,v) ∈ E, in order by nondecreasing wt.

{

if FIND-SET (u) ≠ FIND-SET (v)

{

A ← A ∪ {(u, v)}

UNIONON (u, v)

}

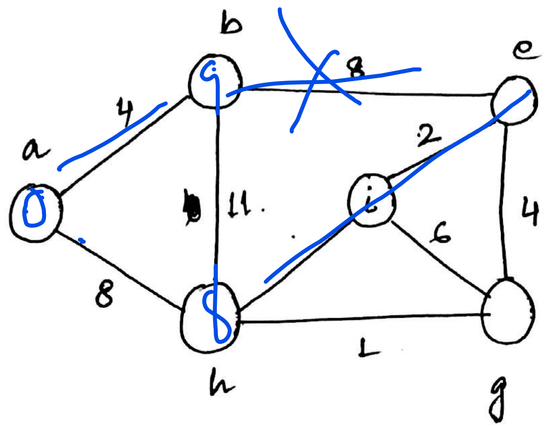
}

return A

PRIMS ALGORITHM

29/03/17

```
MST_PRIMS (G, w, r) {
    for each u ∈ V[G]
    do key[G] ← ∞ , π(u) = nil
    key(r) ← 0        Q ← V[G]

      while (Q != Φ)
      u ← EXTRACT.MIN (Q)
      for each vertex v ∈ V[G].
      {
            if (v ∈ Q AND  w(u,v) < key[Q]))
            {       π (v) ← u
                    key (v) ← w(u, v)
            }

      }
}
```



This determines Path

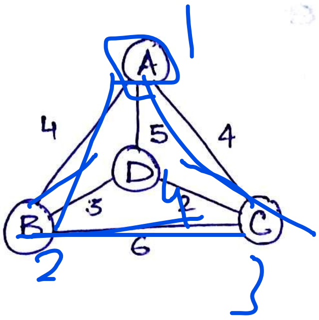
Vertex	π	Key
a	N	0
b	a	4
e	b	8
g	i	4
h	g	2
i	e	2

24/02/17

Travelling Sales Person Problem (TSP)

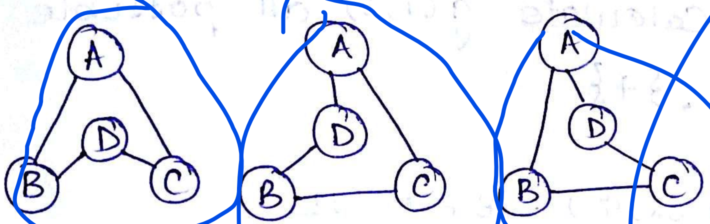
Prob: To find shortest possible route that a salesperson visit each ~~person~~ city exactly once and return to the original city ≪ HAMILTONIAN CYCLE ≫

consider the graph.



W	A	B	C	D
A	0	4	4	5
B	4	0	6	3
C	4	6	0	2
D	5	3	2	0

Possible routes :



cost = 13   cost = 18   cost = 17

Ex: (AUTUMN 2015 QUES.)

FIND THE OPTIMAL TOUR OF THE TRAVELLING SALESPERSON PROBLEM WHOSE ADJACENCY MATRIX IS GIVEN AS FOLLOWS :

W	1	2	3	4
1	0	5	10	15
2	4	0	5	6
3	2	8	0	7
4	10	4	5	0

$g(i, s)$ = length of the shortest path.

$$= \min_{j \in s} [c_{ij} + g(j, s - \{j\})].$$

* Starting at vertex i going through all vertices (s) and terminating at vertex 1.

Step 1: with $[S] = \phi$, calculate

$g(i, \phi)$

$g(2, \phi) = C_{21} = 4$

$g(3, \phi) = C_{31} = 1$

$g(4, \phi) = C_{41} = 10$

Step 2 with $n[S] = 1$ Calculate $g(i, s)$ all possible S are $\{2\}, \{3\}, \{4\}$

$g(3, \{2\}) = C_{32} + g(2, \phi) = 8 + 4 = 12$

$g(2, \{3\}) = C_{23} + g(3, \phi) = 5 + 2 = 7$

$g(2, \{4\}) = C_{24} + g(4, \phi) = 6 + 10 = 16$

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

$g(4, \{3\}) = C_{43} + g(3, \phi) = 5 + 2 = 7$

$g(3, \{4\}) = C_{34} + g(4, \phi) = 7 + 10 = 17$

Step 3 Consider $|S| = 2$ / $n[S] = 2$

Possible sets, $\{2,3\}$ $\{3,4\}$ $\{2,4\}$.

$$g(4,\{2,3\}) = \begin{cases} C_{42} + g(2,\{3\}) = 4+7 = 11 \text{ (min)} \\ C_{43} + g(3,\{2\}) = 5+12 = 17 \end{cases}$$

$$g(3,\{2,4\}) = \begin{cases} C_{32} + g(2,\{4\}) = 8+16 = 24 \\ C_{34} + g(4,\{2\}) = 7+8 = 15 \text{ (min)} \end{cases}$$

$$g(2,\{3,4\}) = \begin{cases} C_{23} + g(3,\{4\}) = 5+17 = 22 \\ C_{24} + g(4,\{3\}) = 6+7 = 13 \text{ (min)} \end{cases}$$

Step 4 $n(S) = 3$.

$\{2,3,4\}$.

$$g(1,\{2,3,4\})$$

$$= \left[\begin{array}{l} C_{12} + g(2,\{3,4\}) = 5+13 = 18 \text{ (min)} \\ C_{13} + g(3,\{2,4\}) = 10+15 = 25 \\ C_{14} + g(4,\{2,3\}) = 15+11 = 26 \end{array} \right.$$

$\therefore$ Minimum cost of tour $= 18$

$\underline{1} \longrightarrow 2 \longrightarrow 4 \longrightarrow 3 \longrightarrow \underline{1}$

Scanned by CamScanner

30|3|17.

GRAPH - TRAVERSAL:

Two ways

a) DFS (Depth First search)

- implemented using stack.

b) BFS (Breadth first search)

- implemented by using queue.

```
a) DFS (V, E)
   {
     for each u ∈ V
     {  color [u] ← WHITE
     }
     time ← 0
     for each u ∈ V
     {  if color [u] == WHITE
        DFS_VISIT (u)
     }
   }

DFS_VISIT (u)
{
   color [u] ← GRAY
   time = time + 1                    # time stamp of vertex u
   d [u] ← time
   for each v ∈ Adj [u]
   {  if color [u] == WHITE
      {     DFS_VISIT (v)
      }
   }
   COLOR [u] ← BLACK                  # end time stamp of u.
   time ← time + 1
   f [u] ← time
}
```

Terminology used :

WHITE ⟵ Node/ vertex not closed.

GREY ⟵ discovered but not finished.

BLACK ⟵ discovered & finished/ visited.

Ex 1. Traverse the following graph by DFS technique with s as start vertex.

i) Draw the DFS tree/ Forest

ii) Find out the DFS sequence

iii) mark ~~the DFS tree edges as Back,~~ forward or cross.

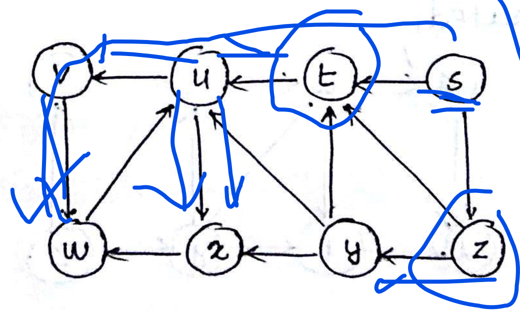
Tree edge ⟶ found by exploring (u,v)

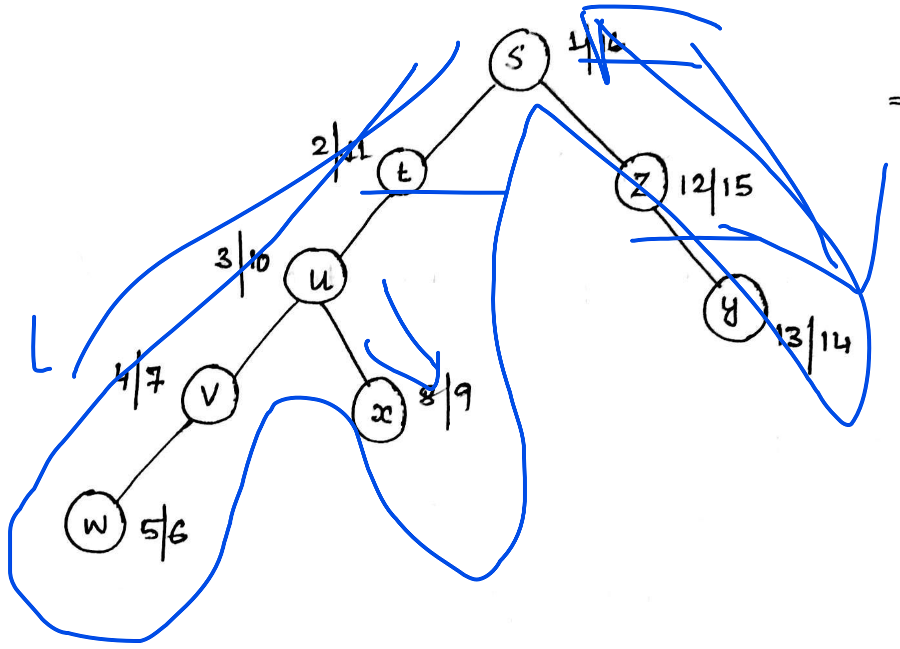
Back edge ⟶ (u,v) where u is a descendant of v

forward edge → (u,v), where u is a descendant of v but not a tree edge.

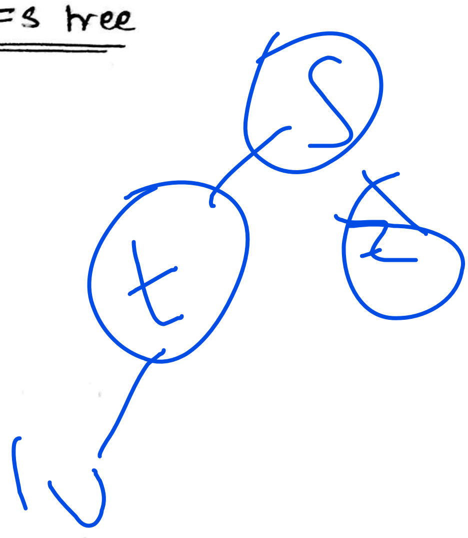
cross edge ⟶ any other edge can go b/w vertices in the same depth first tree or in diff depth first tree.

Given graph.

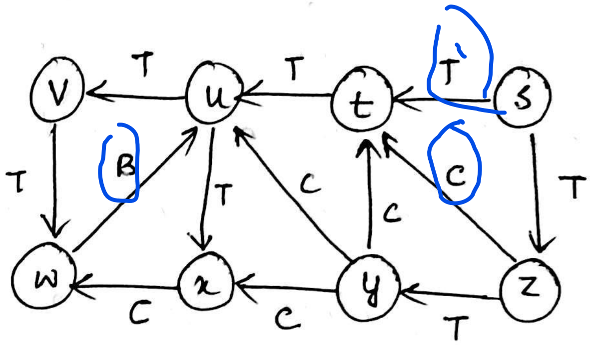




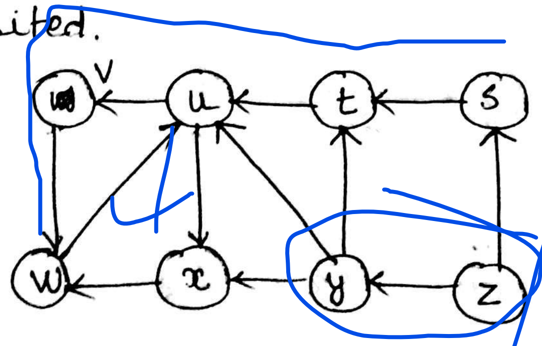
DFS tree

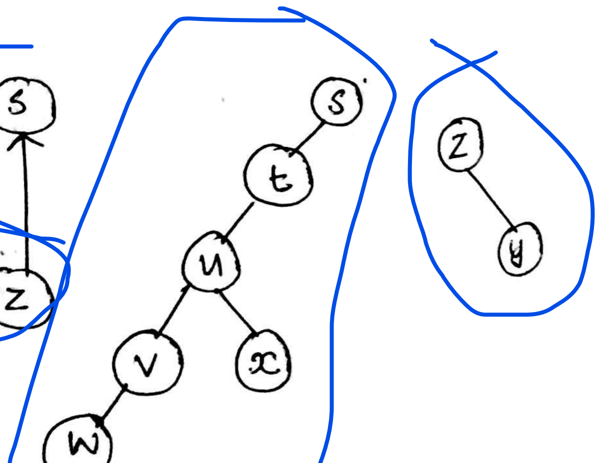


DFS Sequence s, t, u, v, w, x, z, y



Note: create separate ~~forest~~ tree for the nodes not visited.





DFS forest