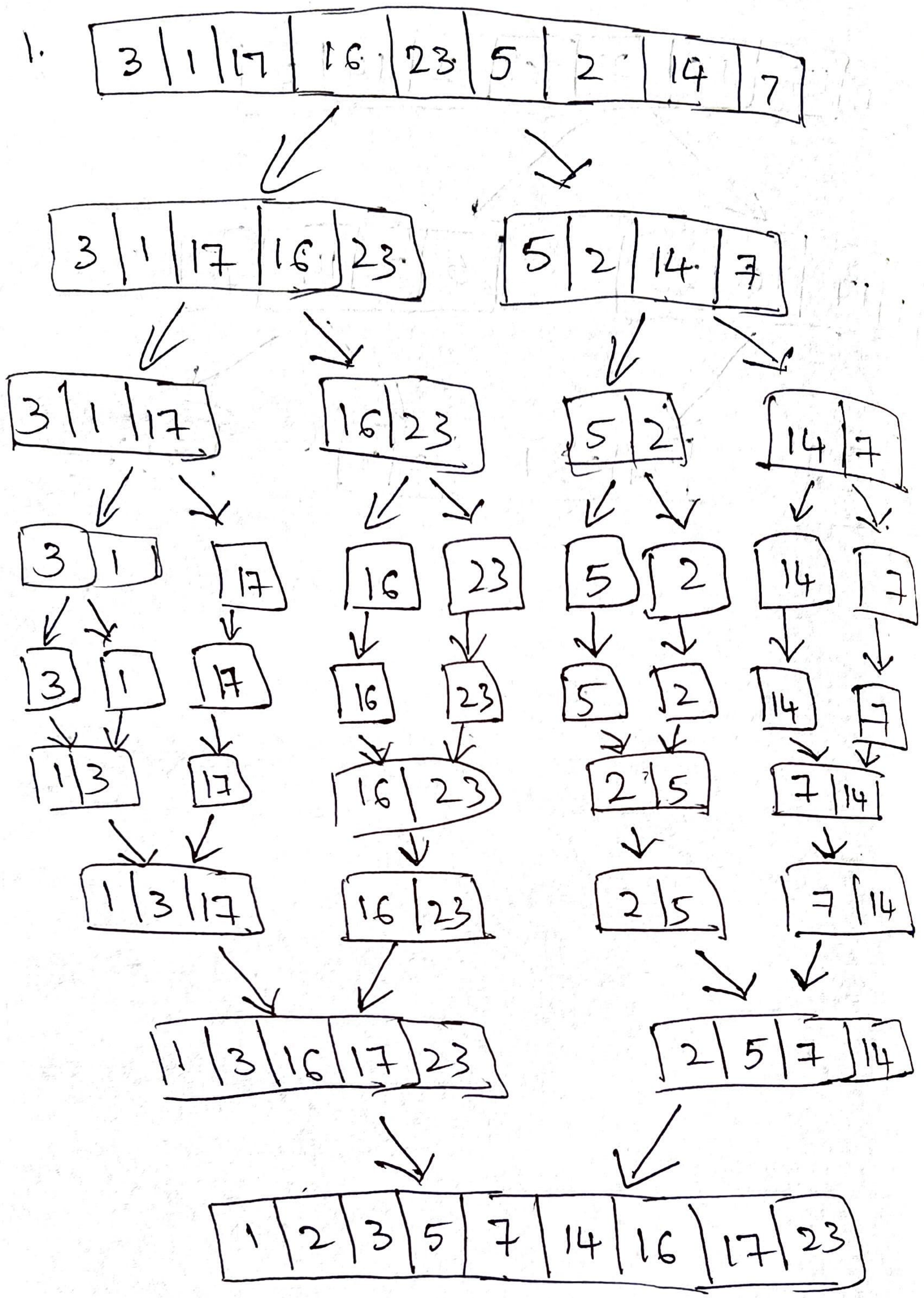
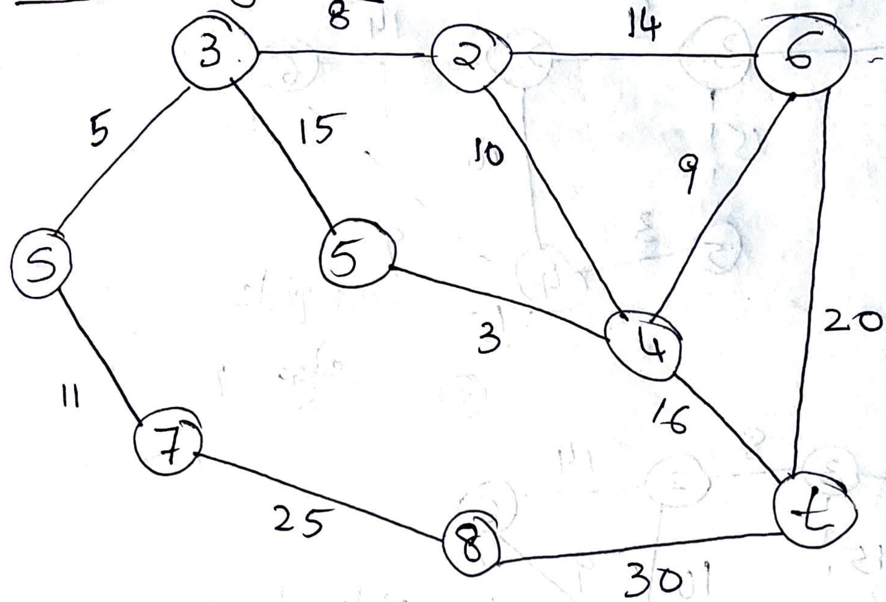
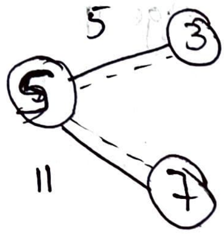
1.


3 | 1 | 17 | 16 | 23 | 5 | 2 | 14 | 7
3 | 1 | 17 | 16 | 23
5 | 2 | 14 | 7
3 | 1 | 17
16 | 23
5 | 2
14 | 7
3 | 1
17
16
23
5
2
14
7
3
1
17
16
23
5
2
14
7
1 | 3
17
16 | 23
2 | 5
7 | 14
1 | 3 | 17
16 | 23
2 | 5
7 | 14
1 | 3 | 16 | 17 | 23
2 | 5 | 7 | 14
1 | 2 | 3 | 5 | 7 | 14 | 16 | 17 | 23

2. Prim's Algorithm -

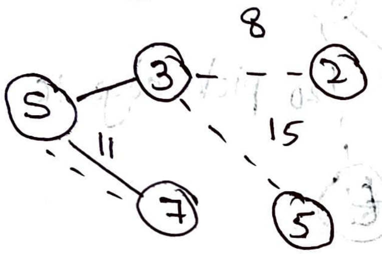


a)



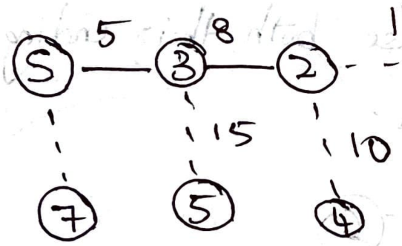
pick edge 5

b)



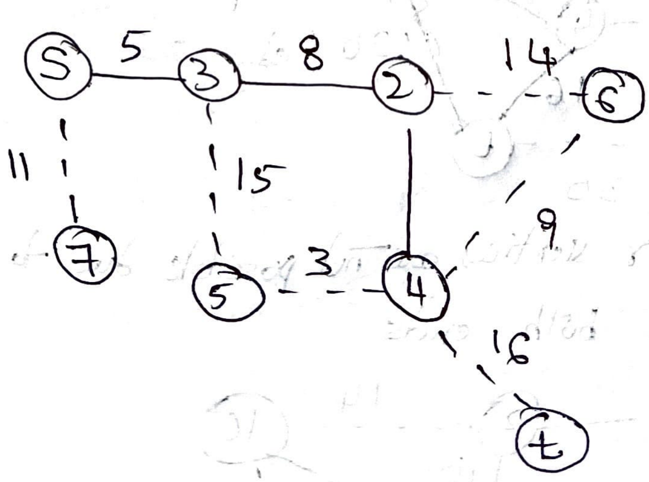
pick edge 8

c)



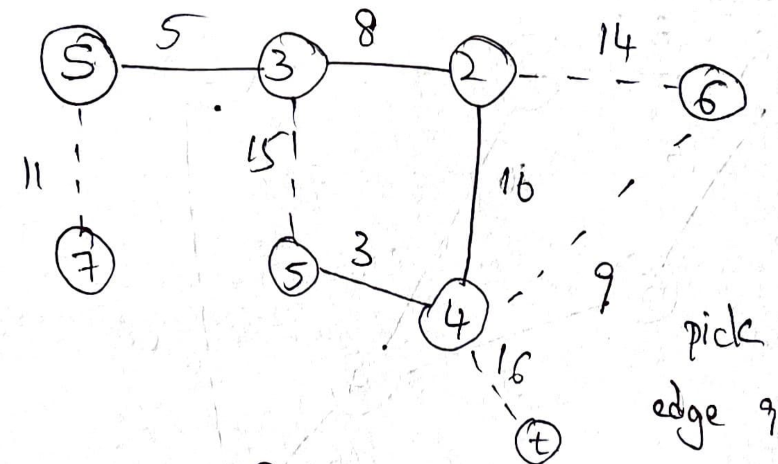
pick edge 10

d)



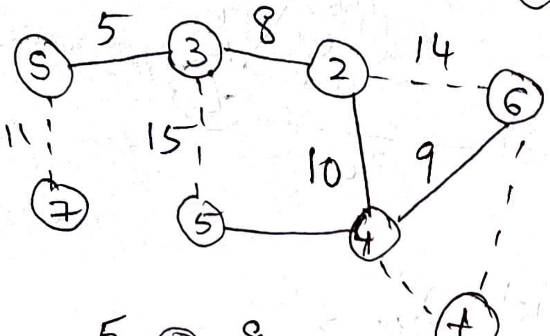
pick edge 3

e)



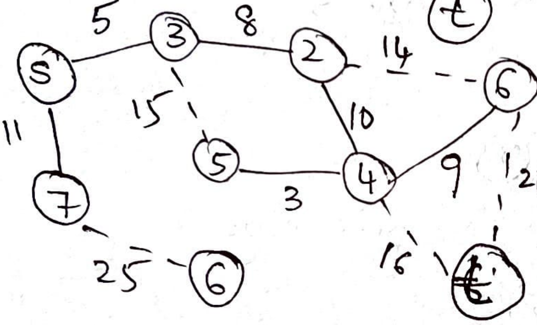
pick edge 9

f)



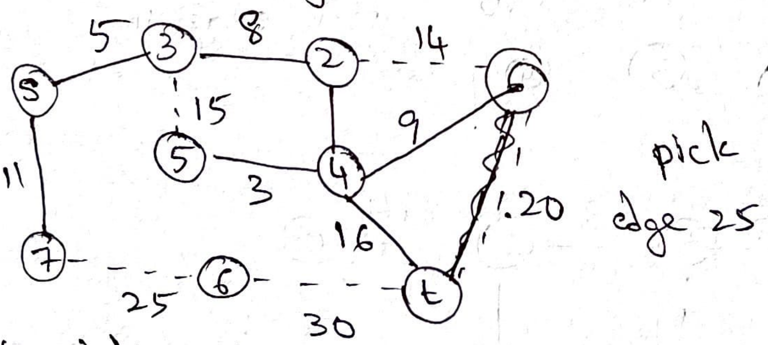
pick edge 11

g)



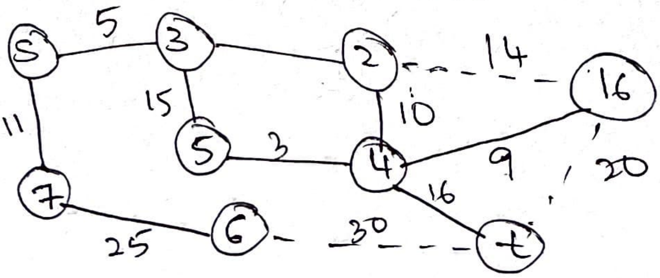
pick edge 16

14,15 are not selected because both their ending vertices are already visited.



pick edge 25

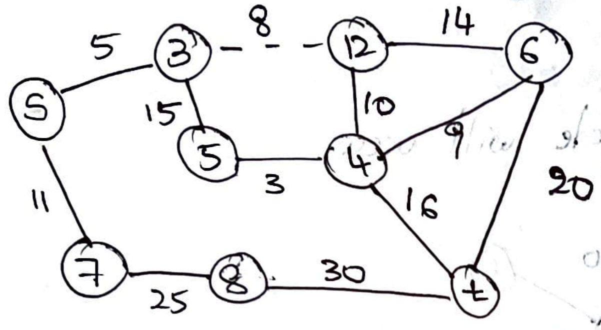
since picking other vertices are not possible due to having visited vertices on both ends



We get the minimal spaning tree because all vertices are now visited

MST weight = 5+8+10+3+9+16+11+25
= 87

Kruskal's Algorithm -



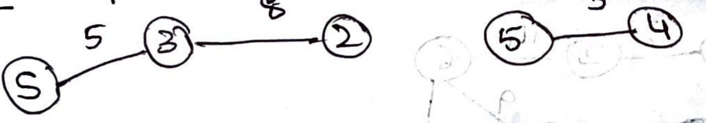
The increasing order of the edges are

→ 3,5,8,9,10,11,14, 15,16, 20, 25, 30

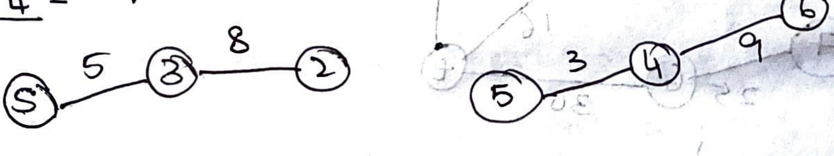
step 1 - pick 3

step 2 - pick 5

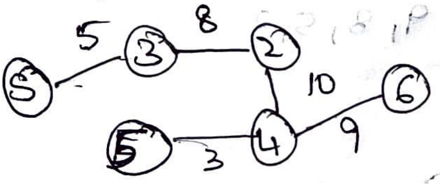
step 3 - pick 8



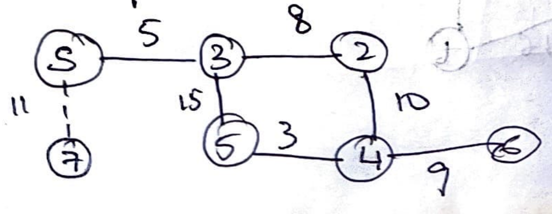
step 4 - Pick 9



step 5 - pick 10

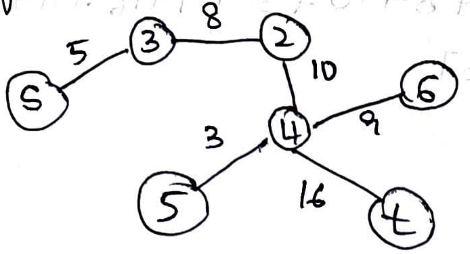


step 6 - pick 11



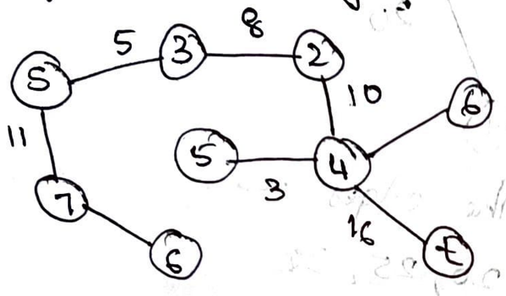
<u>step 7</u> - pick 16

if we pick 14 & 15 cycle will occurs



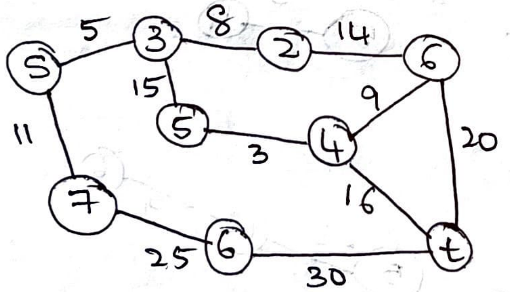
<u>step 8</u> - pick 25

if we pick 20 cycle will occur



all the vertices visited we get minimal spanning tree weight = 87

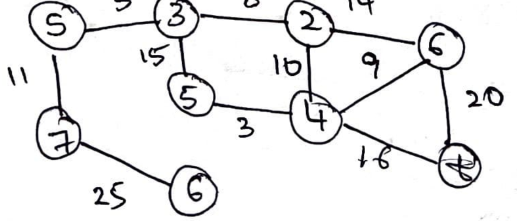
<u>Reverse delete Algorithm</u> -



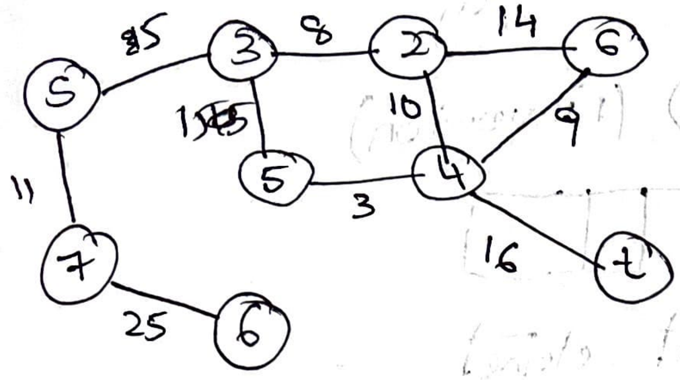
edge of graph in non-increasing order

30, 25, [illegible], 16, 15, 14, 11, 10, 9, 8, 5, 3

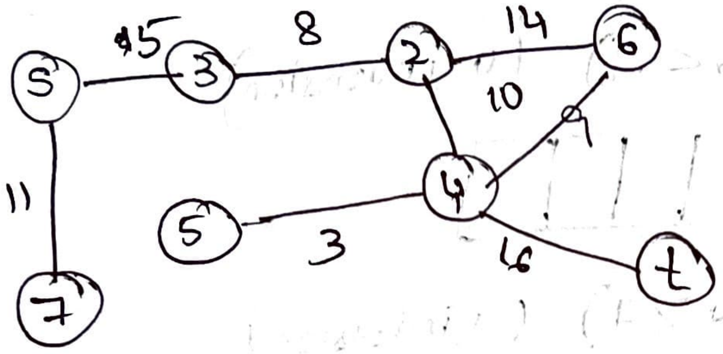
<u>step 1</u> - delete 30



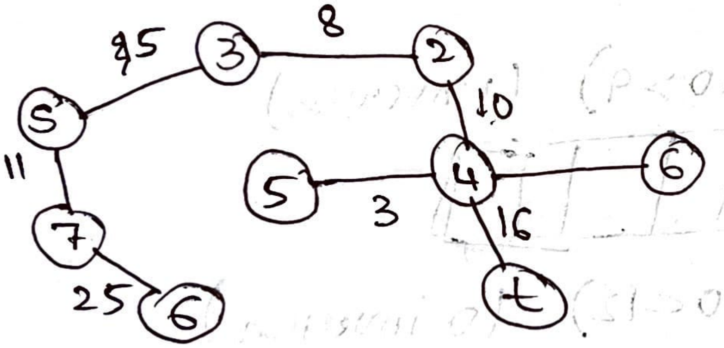
step 2 - for the vertices 25 graph is disconnected delete 20.



step 3 - for the vertices 16 graph is disconnected. delete 15



step 4 delete 14



steps - for the vertices 11, 19, 9, 8, 5, 3 the graph will be disconnected as there are no more edges to delete we have the minimal spanning tree

weight = 87

4.

6	10	15	19	24

a_i

3	5	7	9	12

a_j

$a_i > a_j$ $(6 > 3)$ (1 inversion)

3									

move a_j to next element

$a_i > a_j$ $(6 > 5)$ (1 inversion)

3	5								

$a_i < a_j$ $(6 < 7)$ (0 inversion)

3	5	6							

$a_i > a_j$ $(10 > 7)$ (1 inversion)

3	5	6	7						

$a_i > a_j$ $(10 > 9)$ (1 inversion)

3	5	6	7	9					

$a_i < a_j$ $(10 < 12)$ (0 inversion)

3	5	6	7	9	10				

~~moving a_i or next element~~

3	5	6	7	9	10	15	19	24

$a_i > a_j$ $(15 > 12)$ (1 inversion)

3	5	6	7	9	10	12			

moving a_i as next elements

3	5	6	7	9	10	12	15	19	24

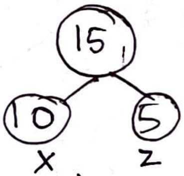
5.

char	frequency	fixed code	Huffman code
A	250	0000	01
E	140	0001	010
C	135	0010	011
D	130	0011	0110
F	120	0100	0000
G	90	0101	1000
H	70	0110	0111
M	65	0111	1111
T	60	1000	0111
P	55	1001	0100
Q	30	1010	1100
R	20	1011	01111
X	10	1100	11111
Z	5	1101	011111

step 1 - character frequency.

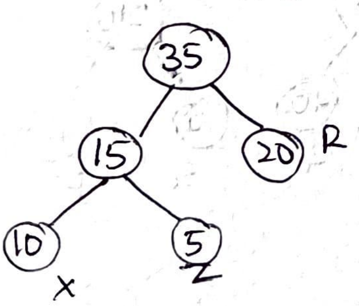
X 10

Z 5



step 2 - character frequency.

R 20

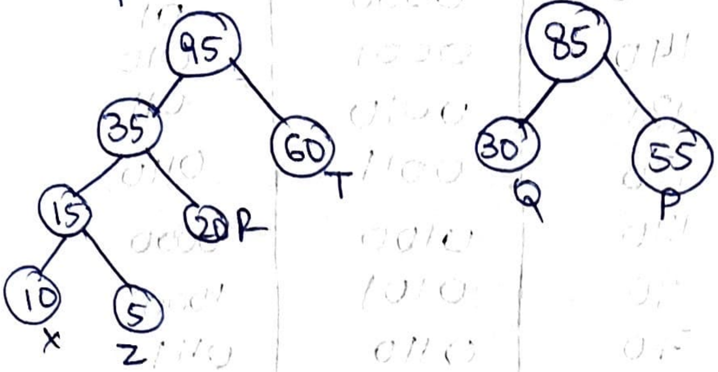


step 3 - character frequency.

P 55

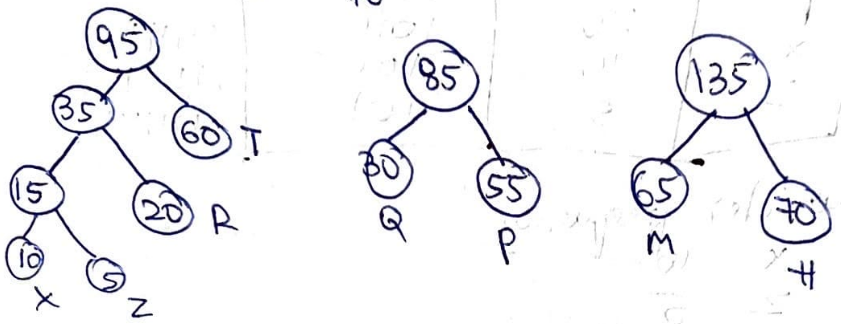
Q 30

step4 - T = 60



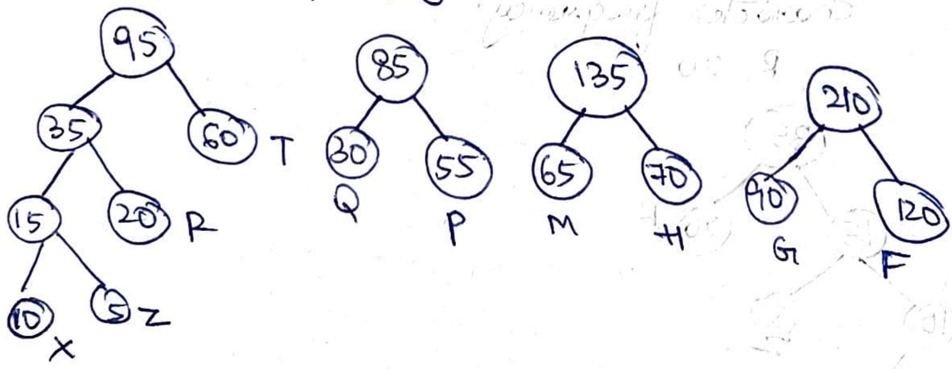
steps - character frequency

M	65
H	70



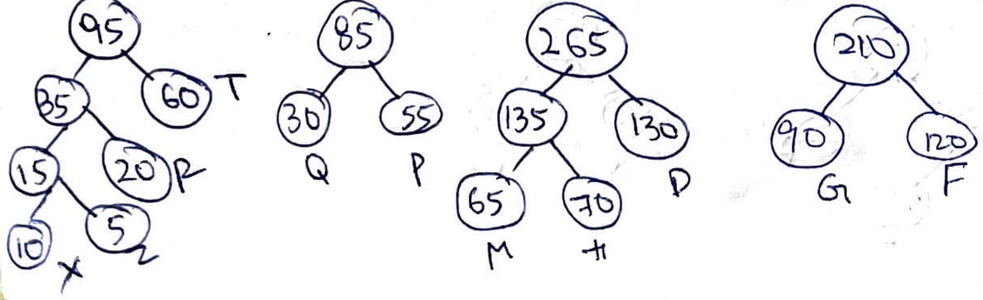
step6 - character frequency

G	90
F	120



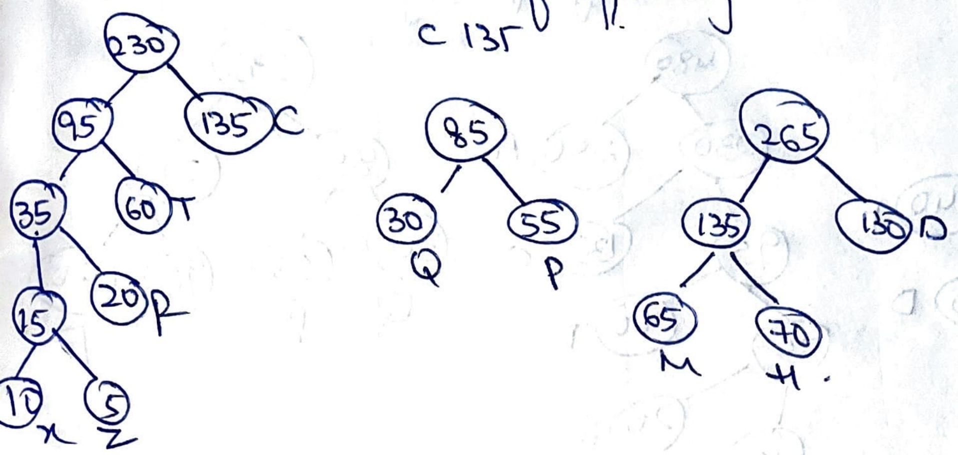
step 7 character frequency

D	130



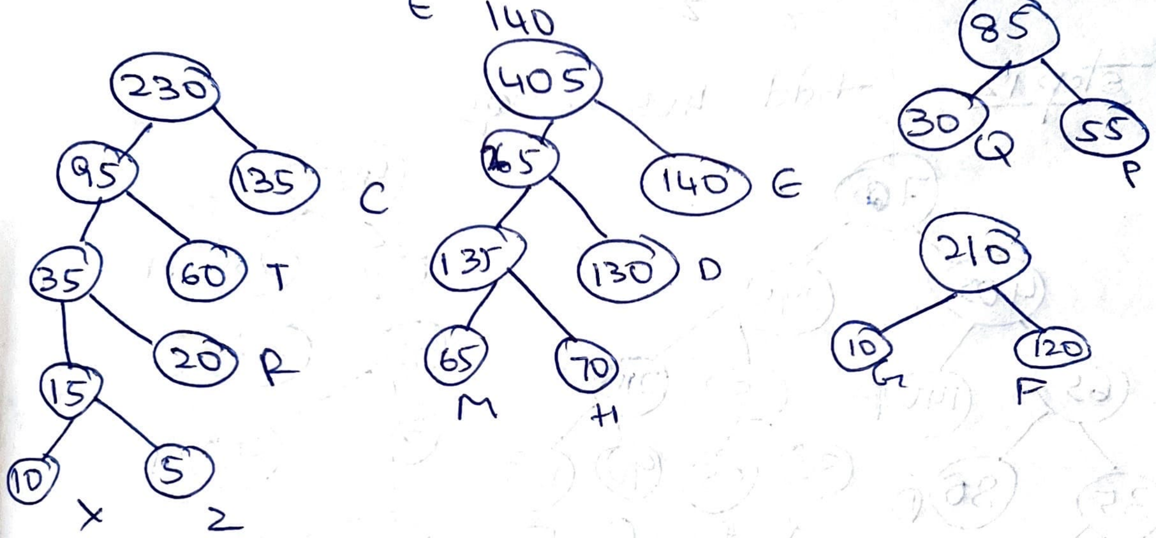
step 8 - character frequency

C 135

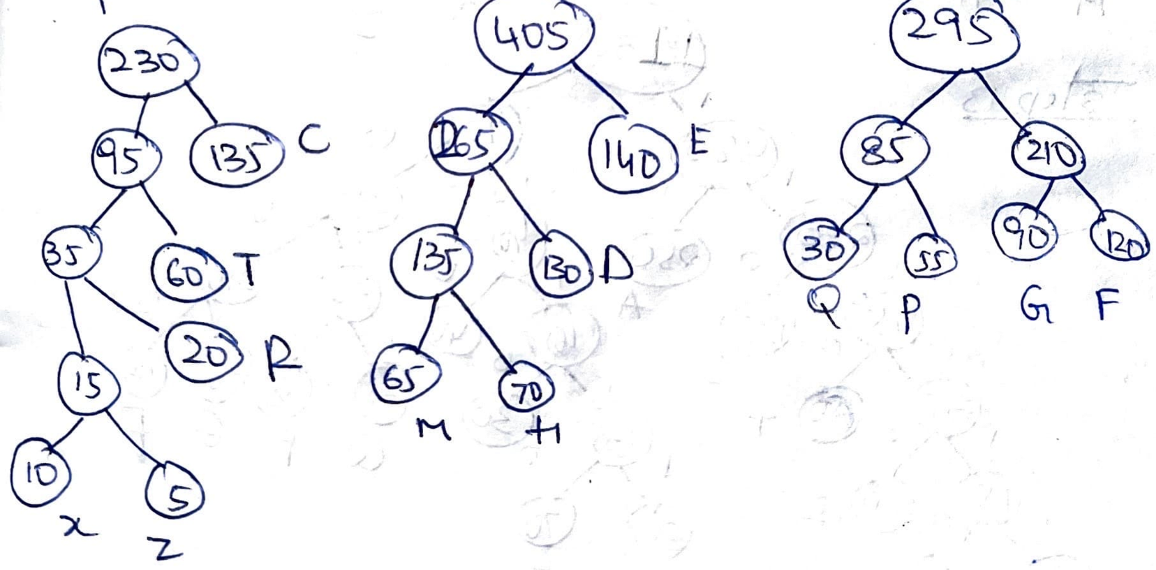


step 9 - character frequency

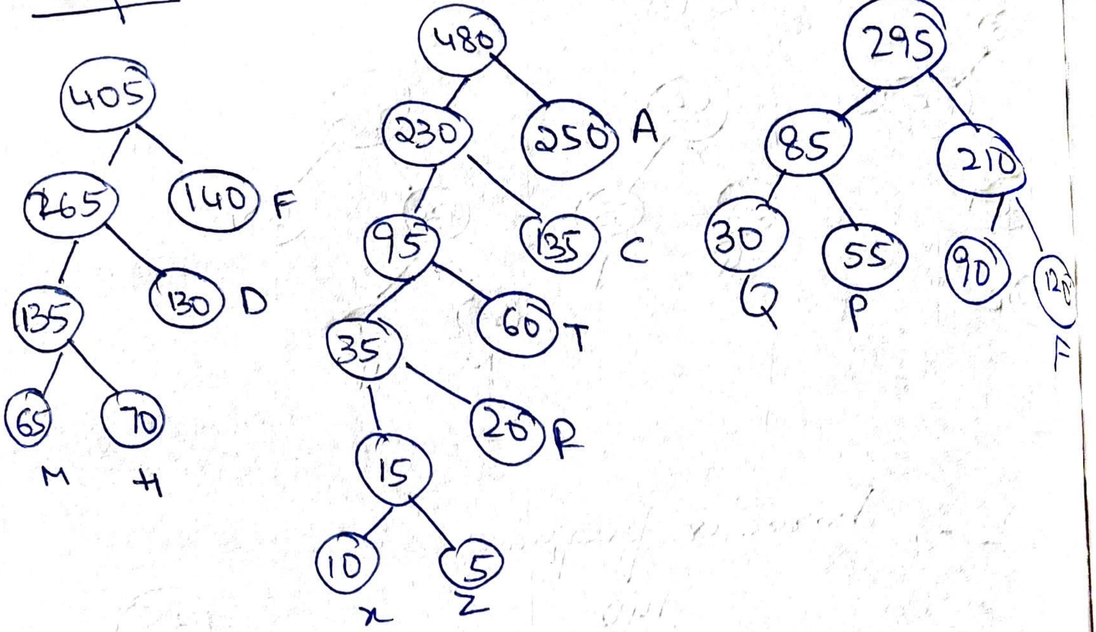
E 140



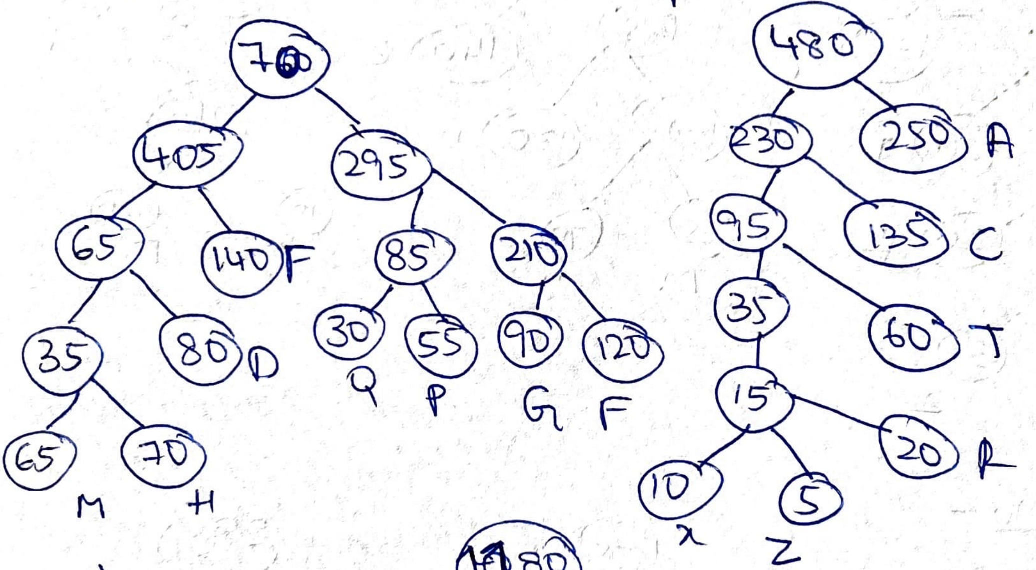
step 10 -



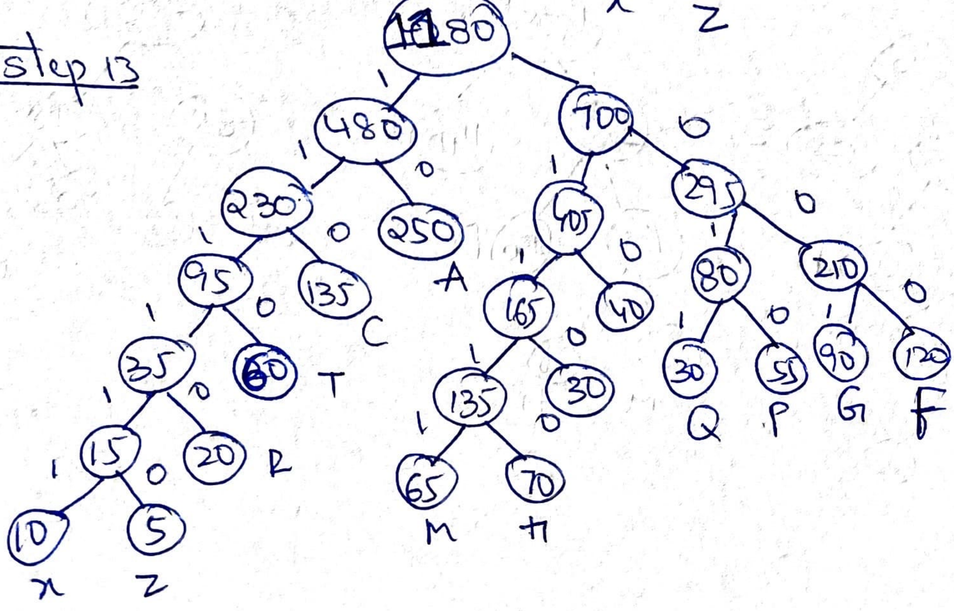
step 11 character frequency:



step 12 - Add 405 & 295



step 13



ABL = $\Sigma f x \cdot depth(x)$ for fixed code

$= (250\times4) + (140\times4) + (135\times4) + (130\times4) + (120\times4) + (90\times4) + (70\times4) + (65\times4) + (60\times4) + (55\times4) + (30\times4) + (20\times4) + (10\times4) + (5\times4)$

$= 4720/1180$

$= 4$

ABL for Huffman coding -

$= \Sigma$ (frequency $\times$ code length) $/ \Sigma$ frequency.

$= (250\times2) + (140\times2) + (135\times2) + (130\times2) + (120\times2) + (90\times2) + (70\times2) + (65\times2) + (60\times2) + (55\times2) + (30\times2) + (20\times2) + (10\times2) + (5\times2)$

$= 2360$

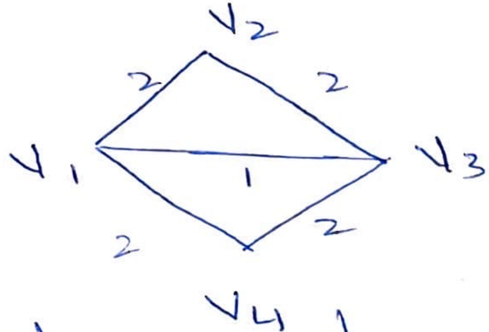
ABL huffman coding $= 2360/1180$

$= 2$

ABL $= 2360/14$

$= 168.57.$

3.



Counter example

~~The edge set $V_1V_2, V_2V_3, V_3V_4, V_4V_1$ are not MST~~

~~the edge V_1V_3 is MST~~

Here the edge set (V_1V_2, V_2V_3, V_3V_4), (V_2V_3, V_3V_4, V_4V_1), (V_3V_4, V_4V_1, V_1V_2), (V_4V_1, V_1V_2, V_2V_3) are not MST but (V_1V_2, V_1V_3, V_1V_4), (V_1V_3, V_2V_3, V_3V_4); (V_1V_2, V_1V_3, V_3V_4), (V_1V_3, V_2V_3, V_1V_4) are MST