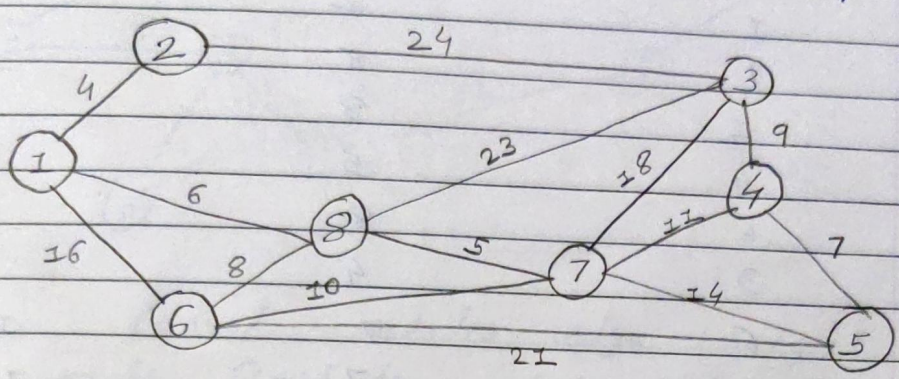
Adv. Algo.

Assignment - 2

2-1

~~Latest~~ Construct the minimum spanning tree:



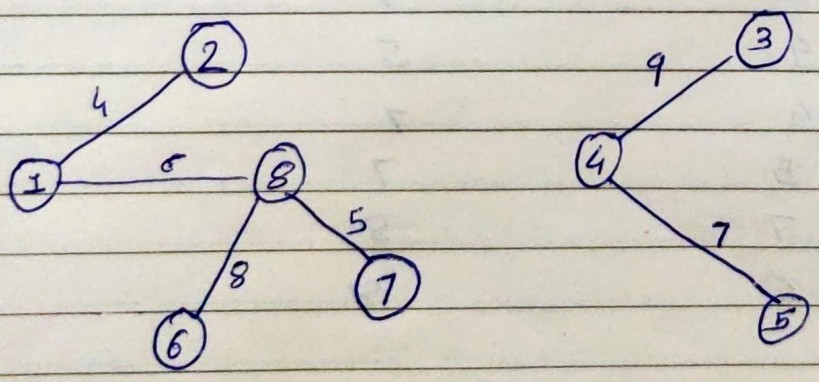
(a) Kruskal's Algo. :-

Node A	Node B	Weight
1	2	4
1	6	16
2	3	24
1	8	6
6	7	10
5	6	21
3	4	9
3	8	23
3	7	18
4	5	7
4	7	11
5	7	14
7	8	5
6	8	8

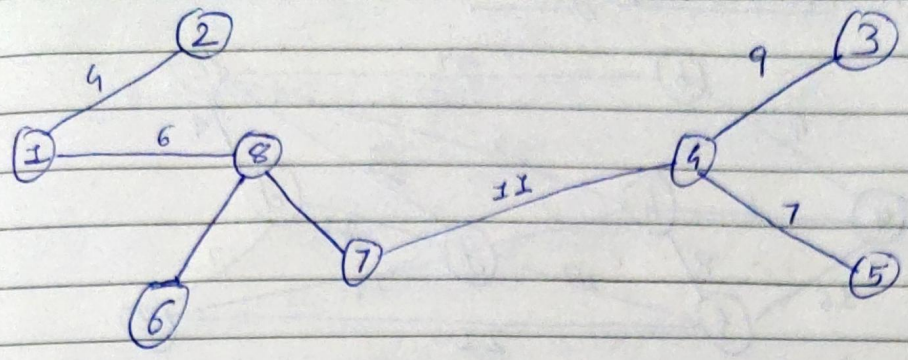
Step - 1 :- Sort all the edges in ascending order of their weight.

Node A	Node B	Weight	
1	2	4	
7	8	5	
1	8	6	
4	5	7	
6	8	8	
3	4	9	
6	7	10	✓
4	7	11	
5	7	14	×
1	6	16	×
3	7	18	×
5	6	21	×
3	8	23	×
2	3	24	×

step-2 :~ Draw all the edges according to the ascending order of their weight & include those edges which do not create a cycle.



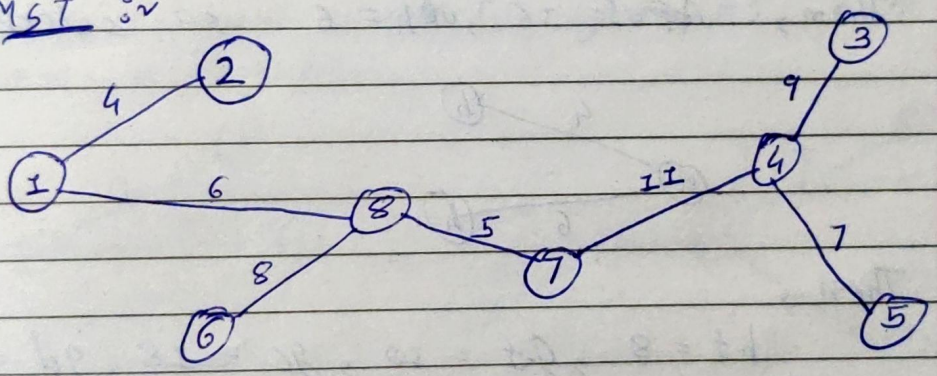
→ Can't take edge 6-7 because it creates a cycle.



→ Cannot includes edge 5-7, it create a cycle. Similarly edge 1-6, 3-7, 5-6, 3-8, 2-3 cannot be included, because these edges create a cycle.

→ No. of edges in MST = $n-1$
$= 8-1 = 7$

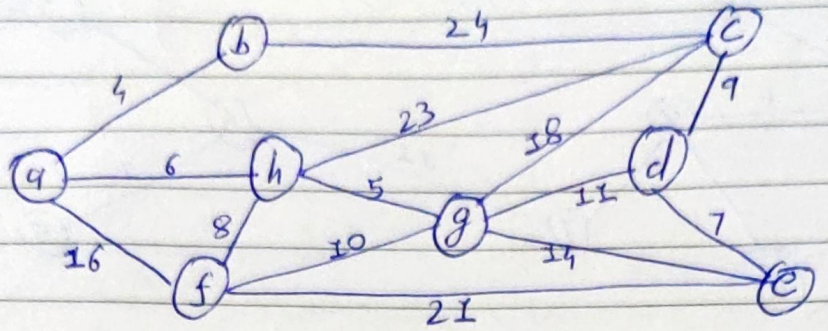
→ MST :-



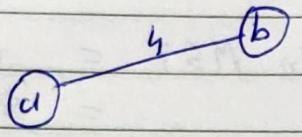
Node A	Node B	Weight
1	2	4
7	8	5
1	8	6
4	5	7
6	8	8
3	4	9
4	7	11

Weight = $4+6+8+5+11+9+7 = 50$

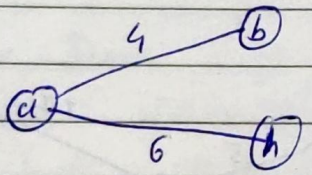
(b) Prim's Algo. :~



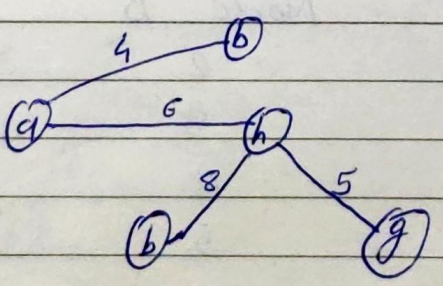
→ Step-I :~ Pick a node,
Suppose, node a, ~~ab = 4~~
then, chose ab = 4. As it has the minimum weight (∵ af = 16 & ah = 6).



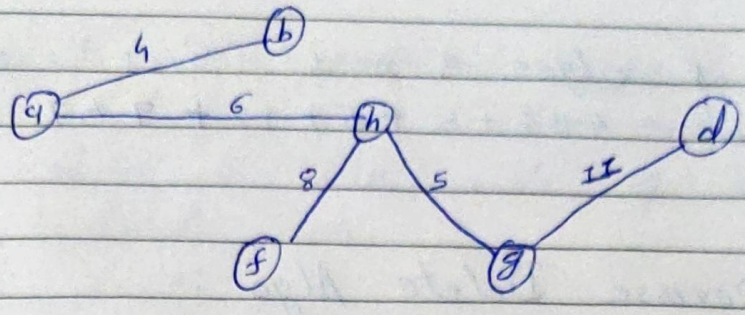
Then, ~~af~~ af = 16, ah = 6 so select ah



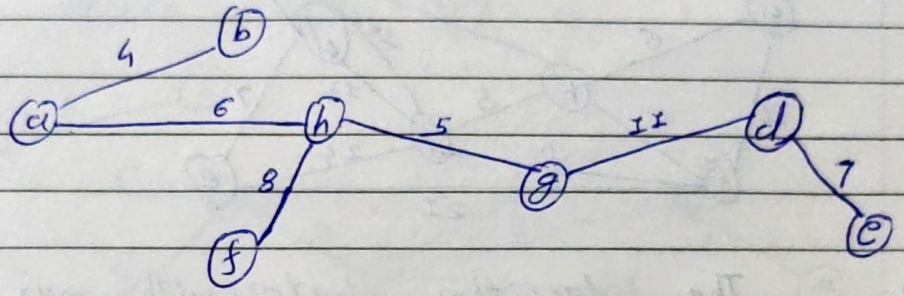
Then,
hf = 8, gf = 10, gc = 18, gd = 11
Select hf = 8



Then,
gf = 10, gd = 11, gc = 18, ge = 14.
Cannot select gf 'coz it creates a cycle.
So, select gd = 11.

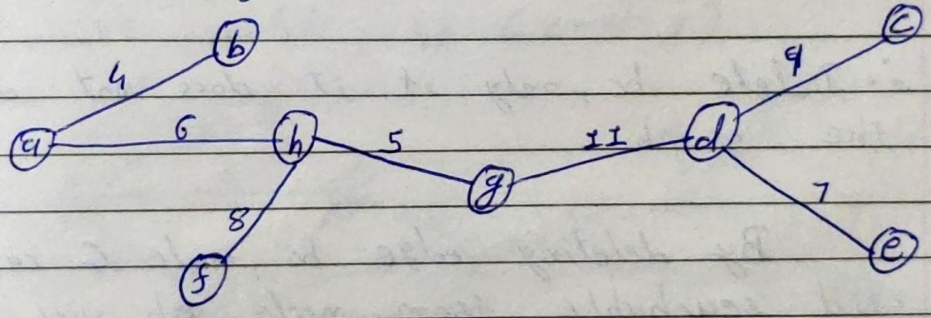


de = 7 , dc = 9 So, select de.



eg = 14 , ef = 21 ,

Cannot select eg & ef both as they form a cycle.

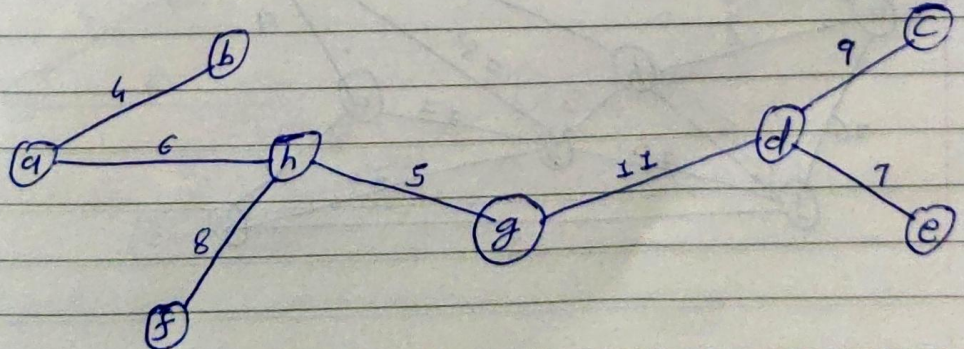


Then,

cg = 18 , cb = 24 , ch = 23 .

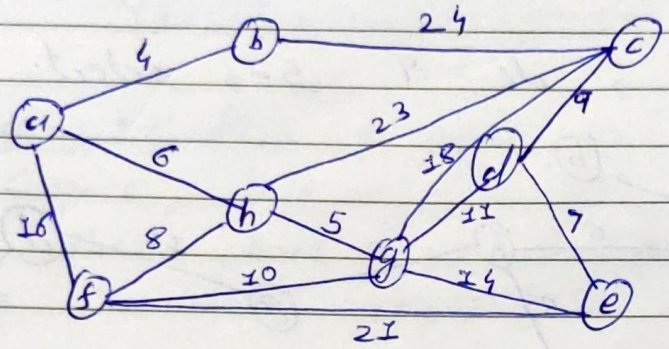
Not able to select any of these edges, as all the three edges, forms cycle.

→



No. of edges = n - 1 = 8 - 1 = 7.
Weight = 4 + 6 + 8 + 5 + 11 + 9 + 7 = 50.

(c) Reverse Delete Algo. :-



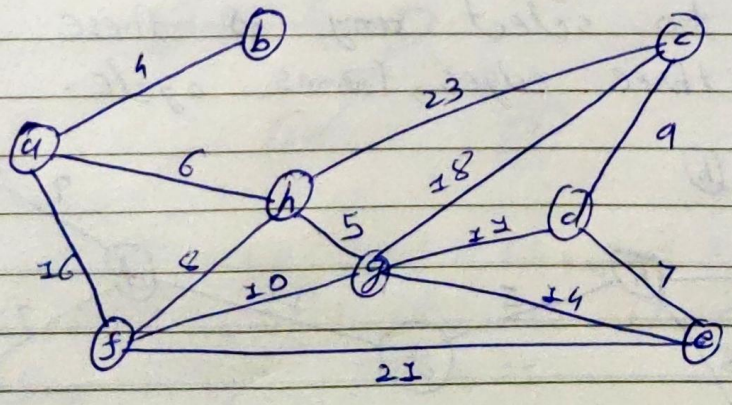
→ The algorithm, starts with max. weighted edge in the graph.

→ In this graph bc = 24 is the max. weighted edge.

∴ Delete bc, only if it does not disconnects the graph.

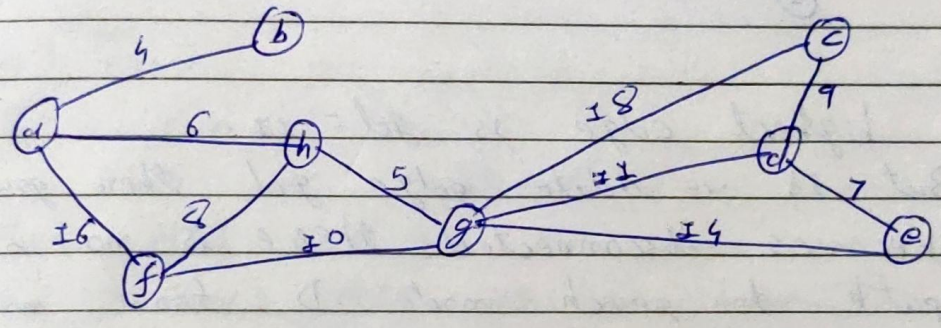
By deleting edge bc, node C can be still reachable from node b via node a and h.

∴ Delete bc.

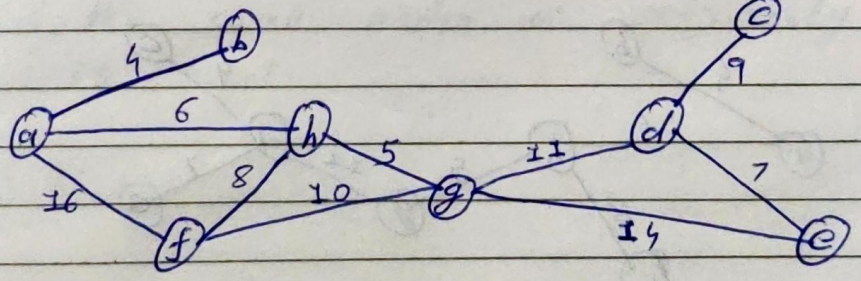


∴ Next max. weighted edge is ch = 23 by deleting this edge. we can reach to node h via node g. (c-g-h)
So, delete ch.

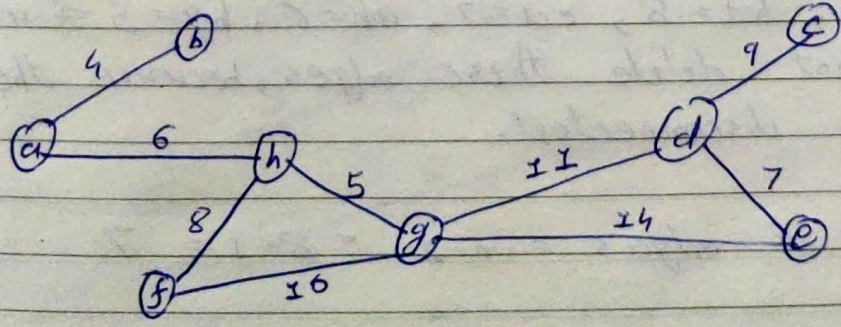
→ Next heighest weight edge is ef = 21. we can reach node f via node g (E-G-F).
So, delete edge ef.



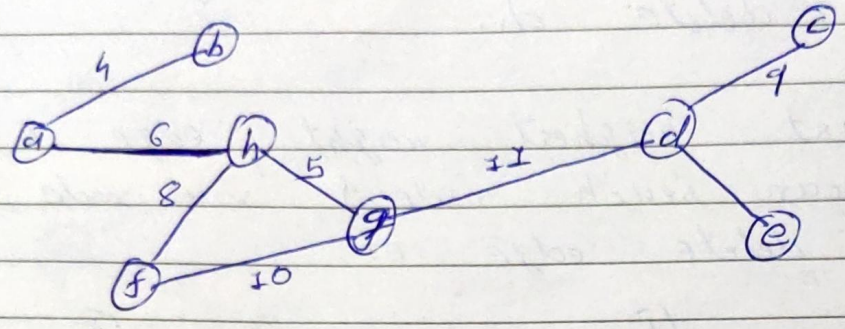
∴ Now, edge cg = 18 the graph is still connected if, we delete cg (c-d-g).



Now, delete edge af = 16.
We can reach node f from node A via node h.

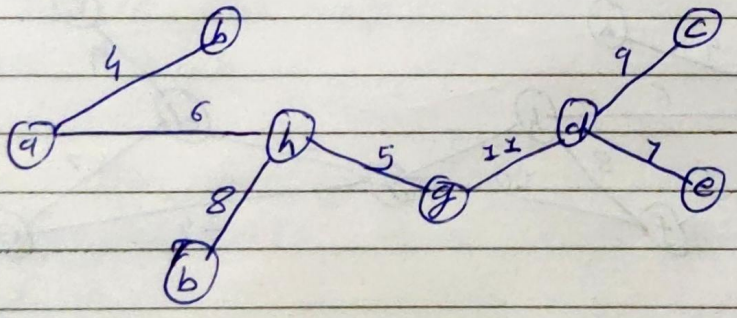


Now, next max. weighted edge is eg = 14 by path e-d-g the graph remains connected, so delete eg.



→ highest edge is gd = 11.
But if we delete edge gd then graph becomes disconnected there is no other path to reach node D from node g.

→ So, fg = 10 connected through path f-h-g.
∴ Delete edge fg.

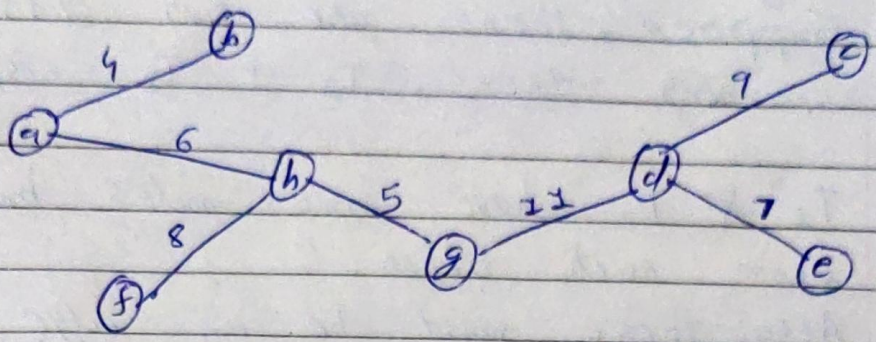


→ Next, me weighted edge cd = 9
But cannot delete cd, graph becomes disconnected. Then, hf = 8, ed = 7, ah = 6, hg = 5 & ab = 4 we cannot delete there edges, because the graph gets disconnected.

→ No. of edges = n-1 = 8-1 = 7

→ Total weight = 4+6+8+5+11+9+7 = 50

→ Final MST :-



Q.2 (a) True.

If $Ce = Ce^2$

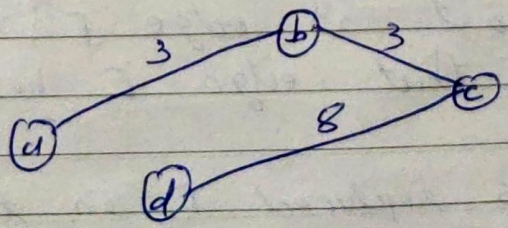
then we construct MST from that graph Then MST will be the same as that of the previous MST as we haven't changed anything just made all the edges squared.

∴ After sorting the squared edges it will be in the same order as previously if Kruskal's algo. is used.

(b) False

In the given graph the shortest path is the edge (c, d)

But, after squaring the costs the shortest path would go through b.



2-3

By Contradiction,
Suppose, there are two different min. spanning trees. T_1 & T_2 respectively.

T_1 & T_2 have some nodes but diff. from each other.
Also, there must be an edge that belongs to any one of them, either T_1 or T_2.

Let this edge be with the min cost & named as E.

Let us assume now that the unique edge E is present in MST T_2.

As TI is also a MST, if edge E is added to the T_1, then a cycle will be from in the T_1 tree.

As tree T_2 is a ~~no~~ MST it do not have any cycles in it Therefore, there must be an edge V which is not present in T_2 but it is present in the cycle formed in tree T_1. when edge E is added.

∴ The cost of edge V should be greater than the cost of edge E, because as stated above that edge E has the min. cost.

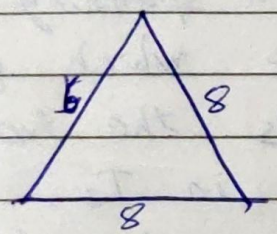
If V is replaced E in the tree T_1, then will create a spanning tree with the least cost.

Thus, by contradiction to our assumption that T_i is also a MST.

Hence, proved that the MST for the connected graph G, whose edge are all distinct will be unique.

Q-4 (a) Every min.-bottleneck tree of G is not a MST of g.

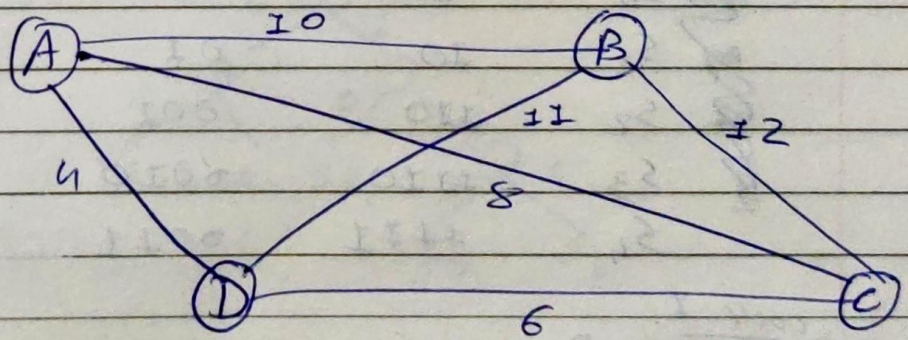
Eg: (Counter Example)



A MST must pick the edge with weight 6.
But a min-bottleneck spanning tree might not pick the edge with weight 6.

A spanning tree T of graph G is a minimum bottleneck tree which considers only the most expensive edge.

eg: ~



For MST of G is
(A, B), (A, C), (C, D).

(b) Suppose,

T_1 = MST

T_2 = Min. bottleneck spanning tree.

$T_1 \neq T_2$.

∴ There is an edge E present in T_1 which is heavier then any other edge in T_2.

By removing edge E from T_1, there must be two connected components.

Therefore As T_2 is a spanning tree, it should have an edge which gets than edge E & connects the two components.

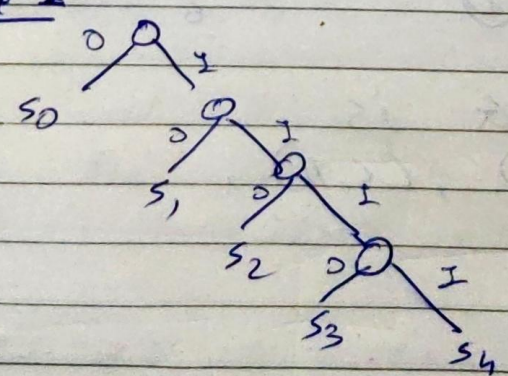
Hence, it is added in T_1.

MST of G is a min. bottleneck spanning tree of G.

Q-5

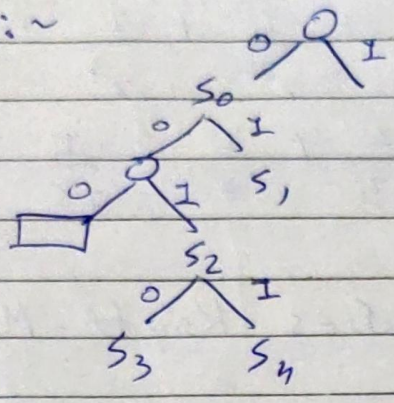
Symbol	Code I	code II	code III	code IV
S_0	0	0	0	00
S_1	10	01	01	01
S_2	110	001	011	10
S_3	1110	0010	111	110
S_4	1111	0011	111	111

(a) code-I



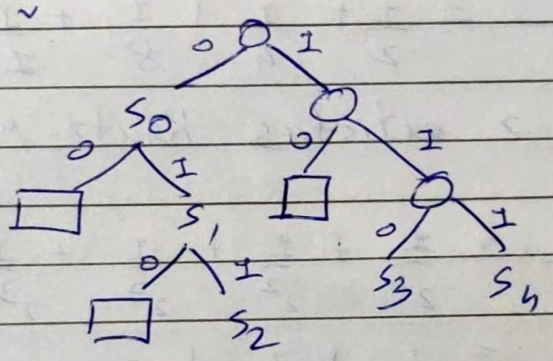
prefix code.

→ code - 2 :~



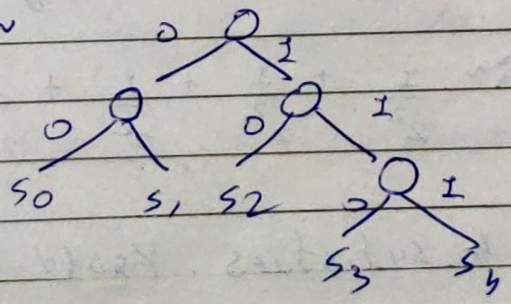
not a prefix code.

→ code - 3 :~



not a prefix code.

→ code - 4 :~



prefix code.

(b) For any uniquely decodable code C,

$$\sum_{i=1} 2^{-l} \leq 1$$

where l is the length of code.

Codes I :- $\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4}$

$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} = \frac{4}{4} = 1$

∴ code-1 satisfies Kraft-McMillan inequality

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

$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{16} = \frac{4}{4} = 1.$

∴ code-2 satisfies Kraft-McMillan inequality.

code-3 :- $\frac{1}{2^1} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^3}$

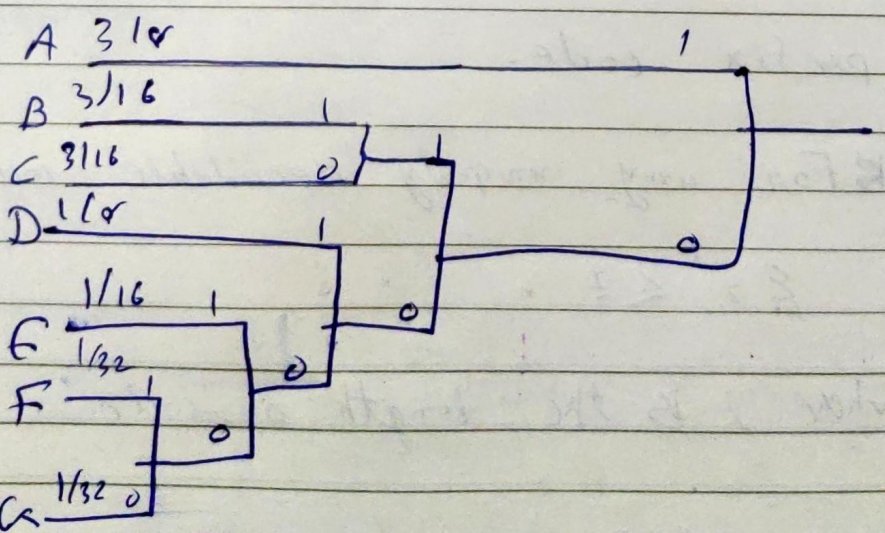
$= \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{8} + \frac{1}{8} = \frac{9}{8} > 1.$

∴ code 3 does not satisfy the inequality.

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

∴ code-4 satisfies Kraft-McMillan inequality.

Q-6

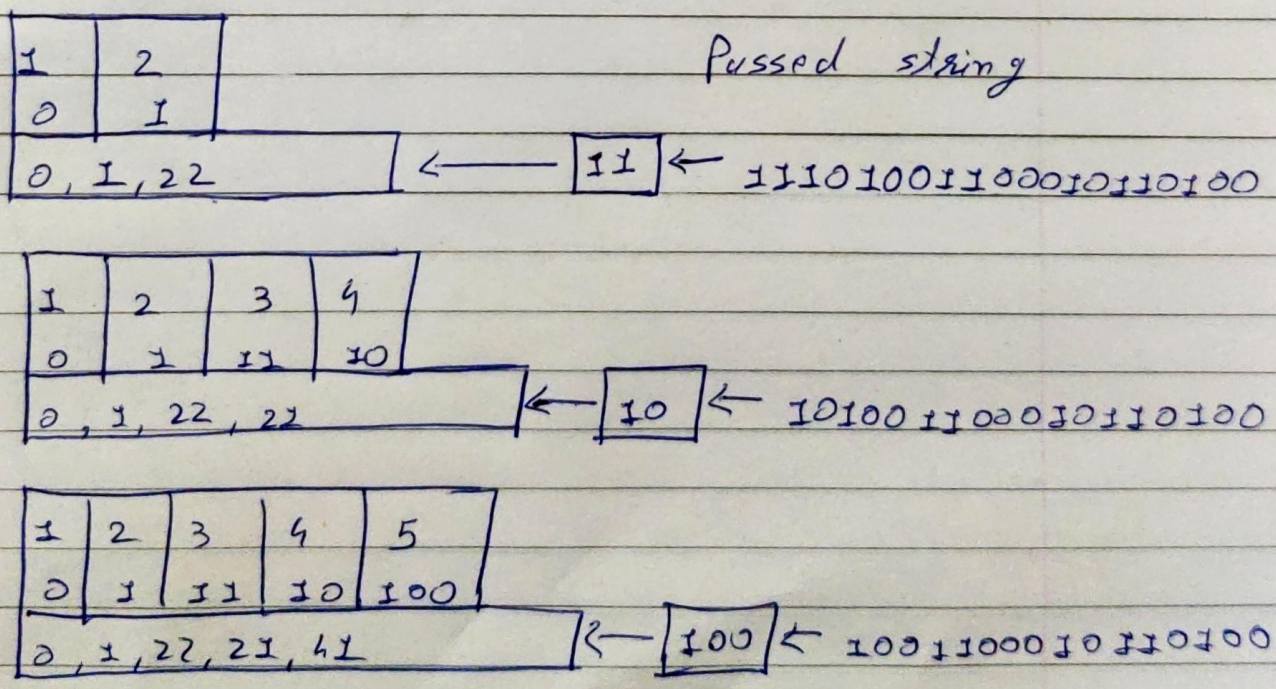


code used for G = 00000
code used for F = 00001
code used for E = 0001
code used for D = 001
code used for C = 010
code used for B = 011
code used for A = 1

Symbol	codeword	length
A	1	1
B	011	3
C	010	3
D	001	3
E	0001	4
F	00001	5
G	00000	5

2-1 11101001100010111010100

Using Lempel-Ziv algo. to encode this seq.



1	2	3	4	5	6
0	1	11	10	100	110

0, 1, 22, 21, 41, 31 ← 110 ← 100110001010

1	2	3	4	5	6	7
0	1	11	10	100	110	00

0, 1, 22, 21, 41, 31, 11 ← 00 ← 00101101100

1	2	3	4	5	6	7	8
0	1	11	10	100	110	00	101

0, 1, 22, 21, 41, 31, 11, 42 ← 101 ← 10110100

1	2	3	4	5	6	7	8	9
0	1	11	10	100	110	00	101	1010

0, 1, 22, 21, 41, 31, 11, 42, 81 ← 1010 ← 10100

0 will not be passed again.

Encoded

Seq. → 0, 1, 22, 21, 41, 31, 11, 42, 81