→ Given that 'm' is the maximum over all vertices, so, by the relax property, we can state that the vertex achieves it's shortest path in v.d.

- By the upper-bound property, after m iterations, no d values will ever change.

- Therefore, no d values will change in (m+1) iteration.

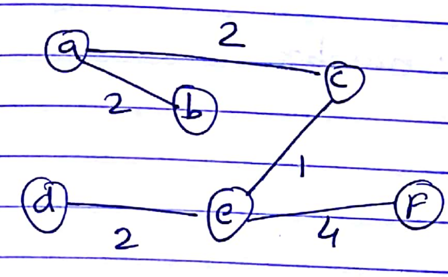
- If we make 'm' iterations, we can't terminate the algorithm.
- Hence, we have to make 'm+1' iterations.

- We will set a flag after the Relax() and check if the edge is relaxed. and pass w.

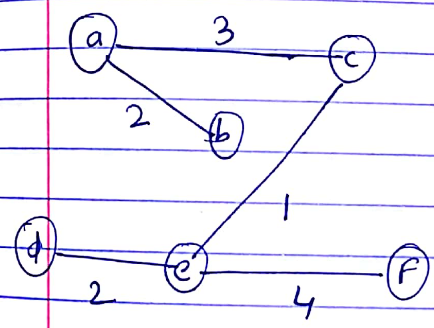
So,
```
Relax (u, v, w)
    if  v.d > u.d + w(u,v)
        v.d = u.d + w(u,v)
        v.prev = u
        visited = true
```

```
BELLMAN-FORD (G, w, s)
  Initialize-single-source (G,s)
  visited = true
  while visited == true          → To take it out
      visited = False                of the while
      for every edge (u,v,w)         loop (m+1)
          Relax (u,v,w)              iterations are
                                     made.
```

for $x=2$

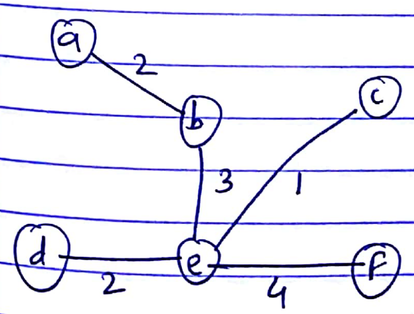


for $x=3$

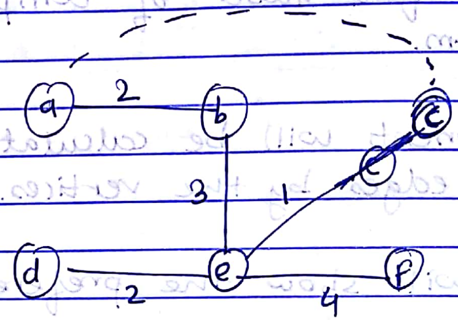


→ if (a,c) is selected

if (a,c) is not selected.



(4) MST of tree T without considering x will be.

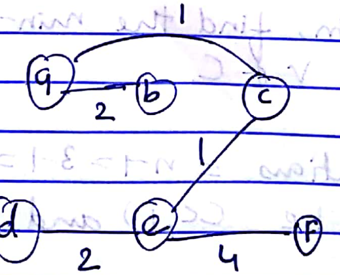


a) Since, all the vertices have been included and we want to keep the MST as T,

if $x > 3$, edge (a,c) won't be selected
∵ it will create a cycle for value of 4
and if $x > 4$, (a,c) won't be selected for MST.

(b) so, if $x \leq 3$ we may get new MST other than T.

For $x = 1$



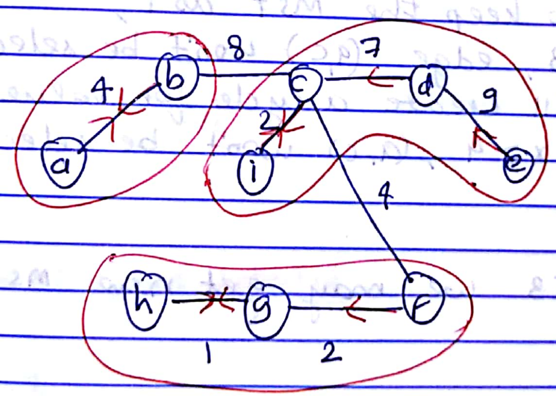
(c) Boruvka's algorithm:

- We need to find number of components for this algorithm.

- These components will be calculated by the selection of edges by the vertices.

- The arrows will show the preference.

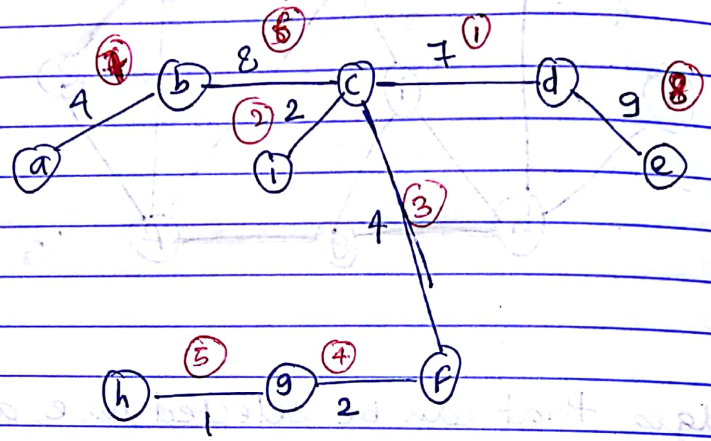
- Non-connected components are independent components.



- Now, according to algorithm, find the min-weight (u,v) with $u \in C$ and $v \notin C$.

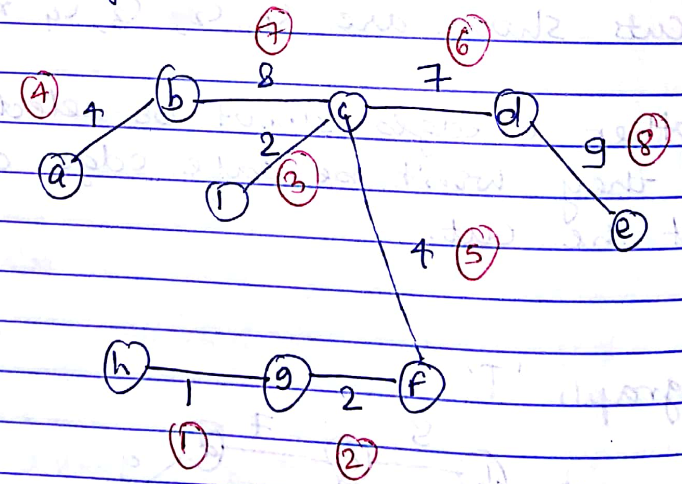
- Therefore, number of iterations $= n-1 = 3-1 = 2$ and edges found will be (c,f) and (b,c) in two iterations.

(a) Prim's algorithm starting with vertex d.



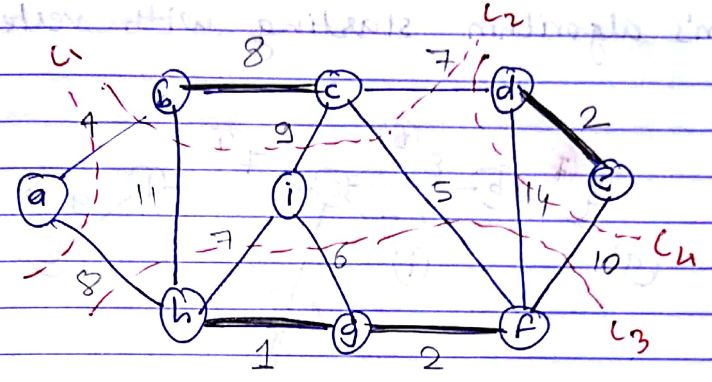
The numbers shown in red color depicts the order of edges selected.

(b) Kruskal's algorithm:



Number depicts the order of edges in which they were selected.

(2)



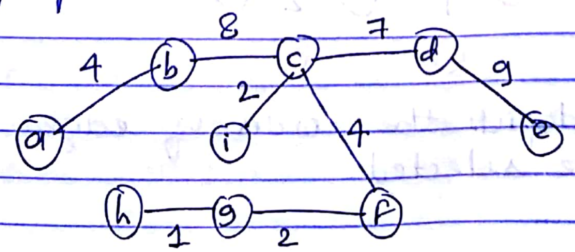
The edges that can be selected as e are :

a) a–b
b) c–f
c) i–g
d) c–d

The cuts shown are c_1, c_2, c_3, c_4 respectively.

– The other 6 edges cannot be selected as 'e' since they won't be safe edge and respect the cut.

(3) Given graph 'T'



$= 5 - 3 + 3 - 3$

$= 2$

$\underline{(A, D)}$

Case 2:

$m = n$.

A = MA<u>LE</u> ; B = <u>L</u>AM<u>E</u>

S = LE.

S.length = 2

$\text{No. of operations} = m - 2 + n - 2$

$= 4 - 2 + 4 - 2$

$= 4$

i) delete M
ii) delete L
iii) insert M
iv) insert L.

Case 3:

$m < n$

~~B = SING~~ ~~SINGER = A~~ , S = SING , ~~S.length = 4~~

$\text{No. of operations} = m - 4 + n - 4$

$= 4 - 4 + 6 - 4$

$= 2 \rightarrow$ delete (E)

$\rightarrow$ delete (R)

(1) ~~A~~ Given,

A.length = |A| = n

and B.length = |B| = m

Task: convert A → B using:

a) delete () → deleting symbol from A.

b) insert () → inserting symbol at any position in A.

Show that:

minimum number of operations for A → B using algo for computing the longest common subsequence.

Let, S be the common longest subsequence of A and B.

Consider three cases:

i) m > n

ii) m = n

iii) m < n

(i) m > n

– when m > n

i.e. B.length > A.length. , S = MAM

eg. MADAM MAM S.length = 3

so, to convert A → B, we have to use insert()

Here, no. of symbols to be inserted will be,

m − S.length + n − S.length