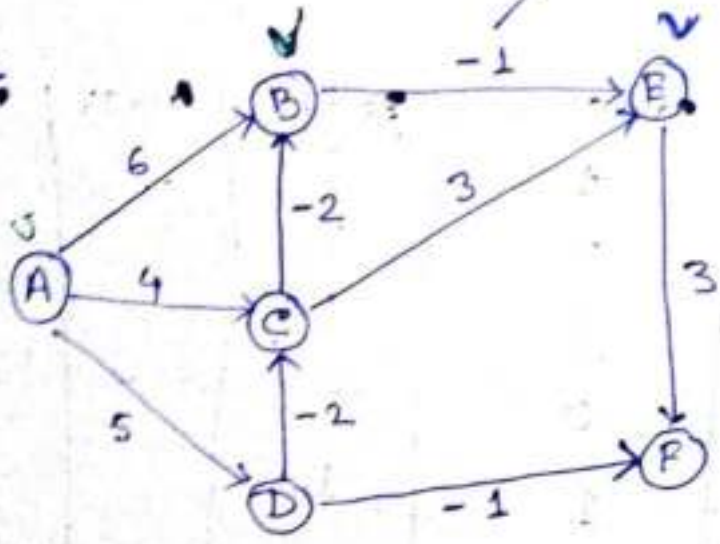
# Bellman-Ford Algorithm

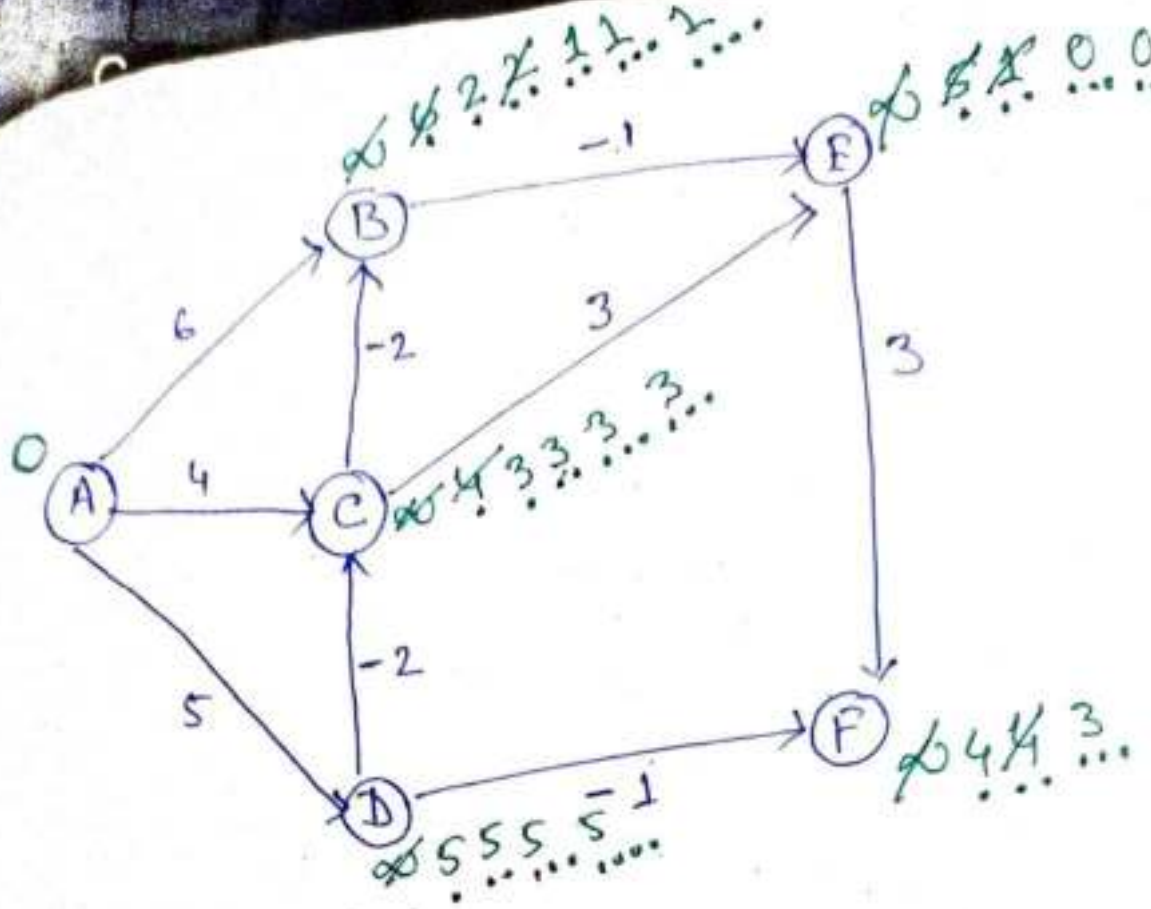
Given a weighted graph $G(V, E)$ with a source vertex $s$, we can solve the single-source shortest paths problem using Bellman-Ford.

The idea is to use Bellman-Ford Algorithm to compute the shortest paths from a single source vertex to all of the other vertices in given weighted digraph. Bellman-Ford algorithm is slower than Dijkstra's algorithm but it is capable of handling negative weights edges in the graph unlike Dijkstra's.

Prob




* Single Source Shortest path
* Set of edges is relaxed exactly $|V|-1$ times, where $|V|$ is the number of vertices in the graph.
* if $(d[u] + c(u,v) < d[v]$
  $d[v] = d[u] + c(u,v))$



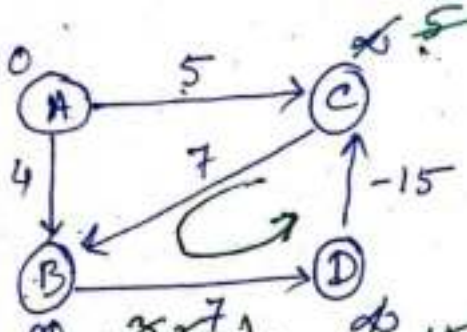

| edges | 1st | 2nd | 3rd | 4th | 5th |
|---|---|---|---|---|---|
| (A, B) → | 2 | 1 | 1 | 1 | |
| (A, C) → | 3 | 3 | 3 | 3 | |
| (A, D) → | 5 | 5 | 5 | 5 | |
| (B, B) → | 5 | 1 | 0 | 0 | same |
| (C, B) → | 2 | 1 | 1 | 1 | |
| (C, B) → | 5 | 1 | 0 | 0 | |
| (D, C) → | 3 | 3 | 3 | 3 | |
| (D, F) → | 4 | 4 | 3 | 3 | |
| (E, F) → | 4 | 4 | 3 | 3 | |

| vertex | |
|---|---|
| A | → 0 |
| B | → 1 |
| C | → 3 |
| D | → 5 |
| E | → 0 |
| F | → 3 |

* Bellman-Ford makes $|E|$ relaxations for every iteration, and there are $|V|-1$ iterations. therefore, the worst case scenario is that bellman-ford runs in $O(|V| \cdot |E|)$ time.

$O(V \cdot E)$

* Bellman-Ford Algorithm will not work if the graph contains any negative weight cycle.




| edges: | 1st Ite | 2nd | 3rd (stop) | 4th |
|---|---|---|---|---|
| A, B → | 4 | 3 | 2 | 1 |
| A, C → | -4 | -5 | -6 | -7 |
| ~~B, A →~~ | ~~0~~ | ~~0~~ | | |
| C, B → | 4 | 3 | 2 | 1 |
| ~~B, C →~~ | ~~0~~ | ~~0~~ | | |
| B D → | 11 | 10 | 9 | 8 |
| D C → | -4 | -5 | -6 | -7 |

No update

Should not be any changes but here change, so, bellman-ford Algo is not giving correct ans in this graph.

# Dijkstra's Shortest Path Algorithm :-

- Step 1 : Initially all nodes are "non-permanent"

S2:- Set the source node (A) as permanent.
A is the same time the 'current node'.

S3:-i) Examine all non-permanent nodes i adjacent to the current node.

ii) For, each i, calculate the cumulative distance from source node to i via the current node.

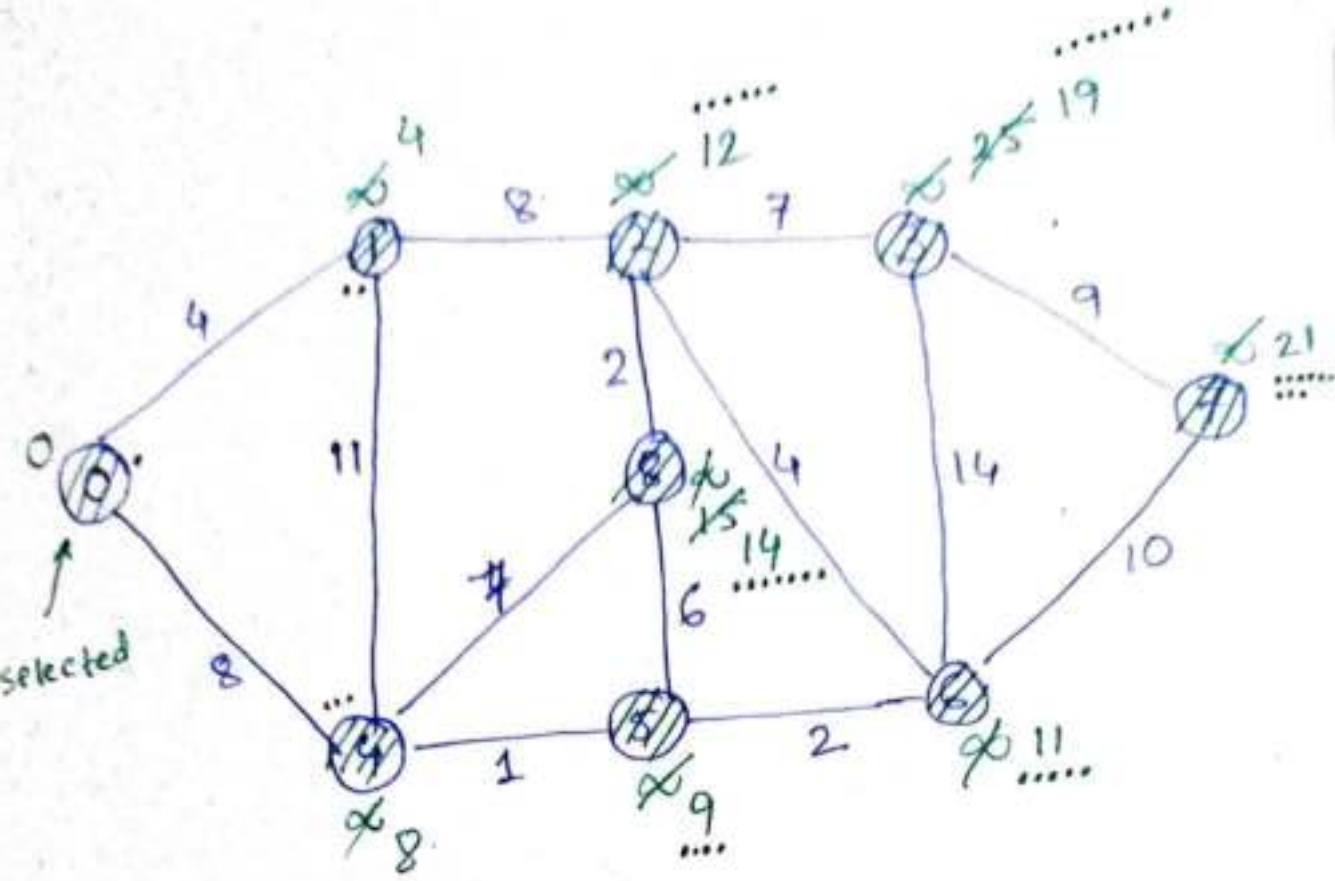
iii) Relabel i with the newly computed distance
- but if i already has a shorter cumulative distance than calculated one, then to not relabel.

iv) Also, label i with name of the current node (as predecessor)

v) compare label (distance) of all non-permanent nodes and choose the one with the smallest value. change the node to permanent and set it as the current node.

vi) Repeat step 3 until all nodes become parment.

* Time complexity :- $O(E \log V)$.




if $(d[u] + c(u,v) < d[v],$
$d[v] = d[u] + c(u,v))$

vertex

0 – 0
1 – 4
2 – 12
3 – 19
4 – 8
5 – 9
6 – 11
7 – 21
8 – 14

# Prim's Algorithm

→ used to draw Minimum spanning tree (MST) is defined for weighted graph).

(MST) → spanning tree of a graph consist of all vertices & some of the edges, so, that the graph does not contain a cycle.

Algorithm :-

Let G(V, E) is a connected, weighted, undirected graph.

1) create two sets V and V', such that
   - V' is empty
   - V contains all vertices of graph.
   - Select minimum weighted edge (i, j) from G and inserts i & j into the set V'.

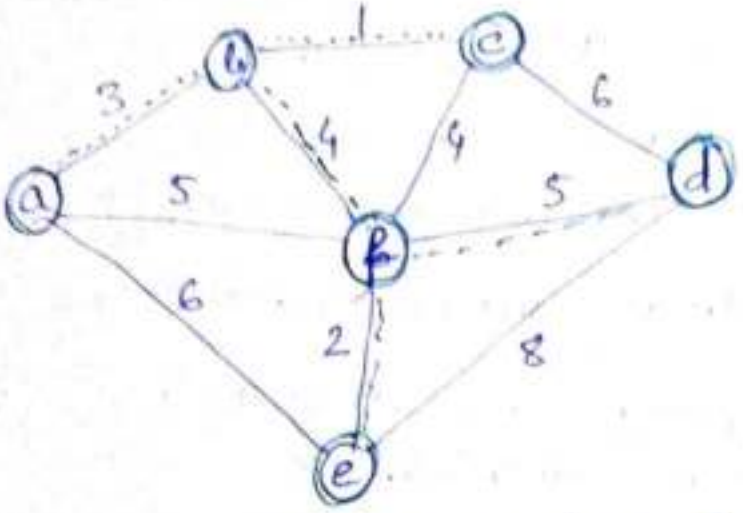
2) Repeat step 3 while V is not equal to V'.

3) Find all neighbors of all vertices which are in set V' such that one end point of neighboring edge is in V' and another not in V'.

4) sum up all selected edge's weight(s), Exit.

* [ Prim's algorithm is use to find minimum cost spanning tree for a weighted undirected graph. ].

Example- 1



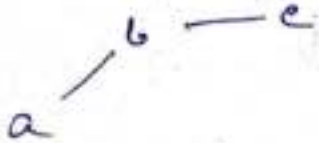

vertices : $\{a, b, c, d, e, f\}$

find out minimum edge in the given graph, and draw, start with

① b —— c

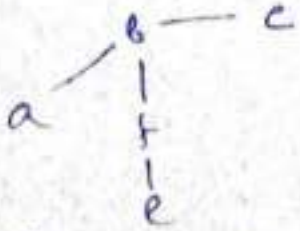
find minimum weight from b, and c ; so choose b to a

②

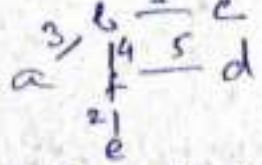



③ now find minimum weight from b, c, a (no cycle), so choose b to f (any one can choose c to f, for same weight)


④ choose minimum, and remember that loop not formed choose f to e




⑤ choose f to d




stop here, because we travers all vertices.

So minimum cost is = 1 + 3 + 4 + 5 + 2

= 15

few points

1) Remove all the loops (any edges that and ends at the same vertex is loop).

2) Remove all parallel edges between two vertices except the one with least weight.

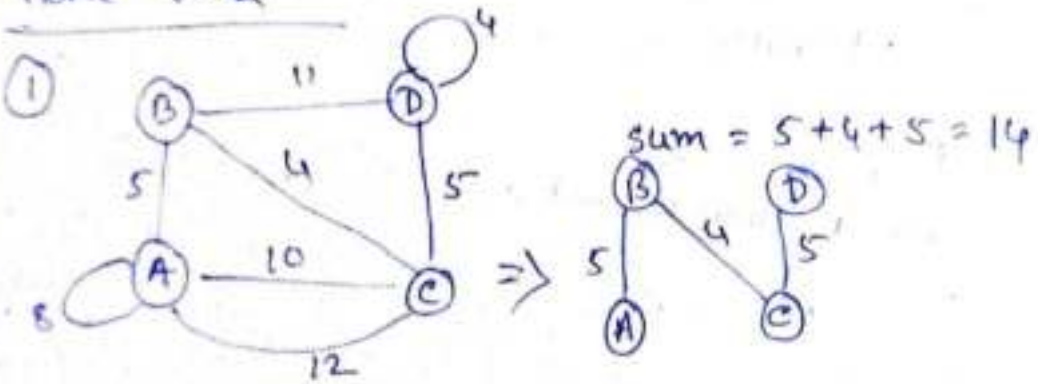
3) choose any arbitary vertex as root vertex

4) check outgoing edges and select the one with less cost and with no cycle.

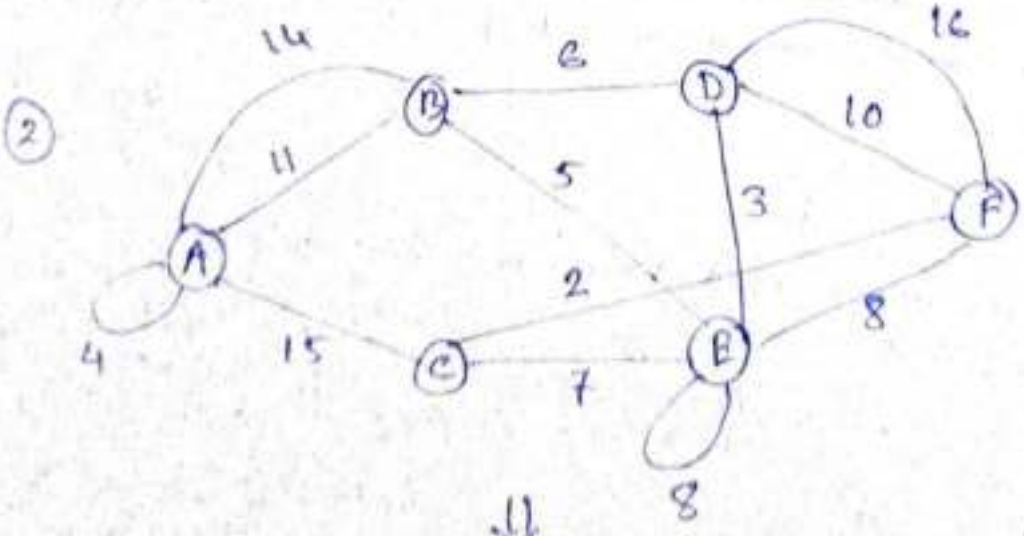
---

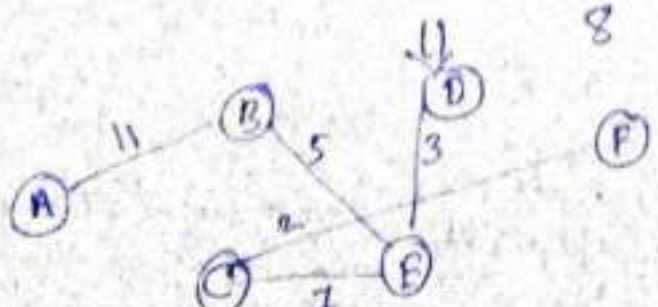
Home task

(1)




(2)




sum = 11 + 5 + 7 + 3 + 2
= 28

# Kruskal's Algorithm

Kruskal's Algorithm is use to find minimum cost spanning tree for a weighted undirected graph.

## Algorithm

Step 1 :- Remove all the loops

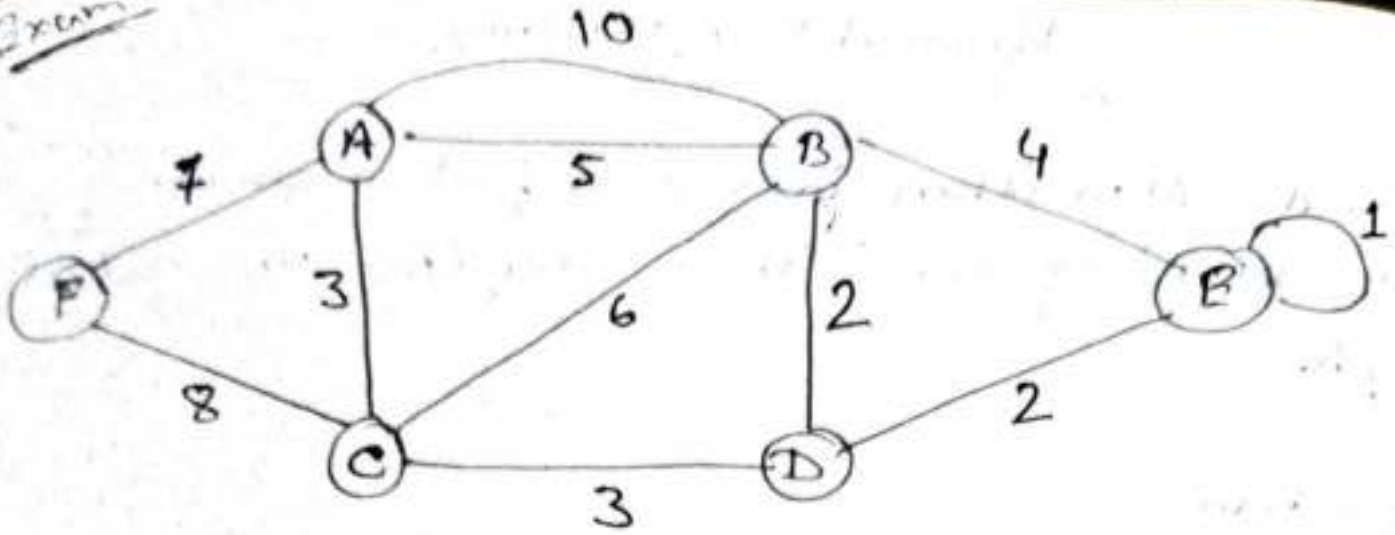
Step 2 :- Remove all parallel edges between two vertices except the one with least weight.

Step 3 :- Arrange all edges in their increasing order of weight.

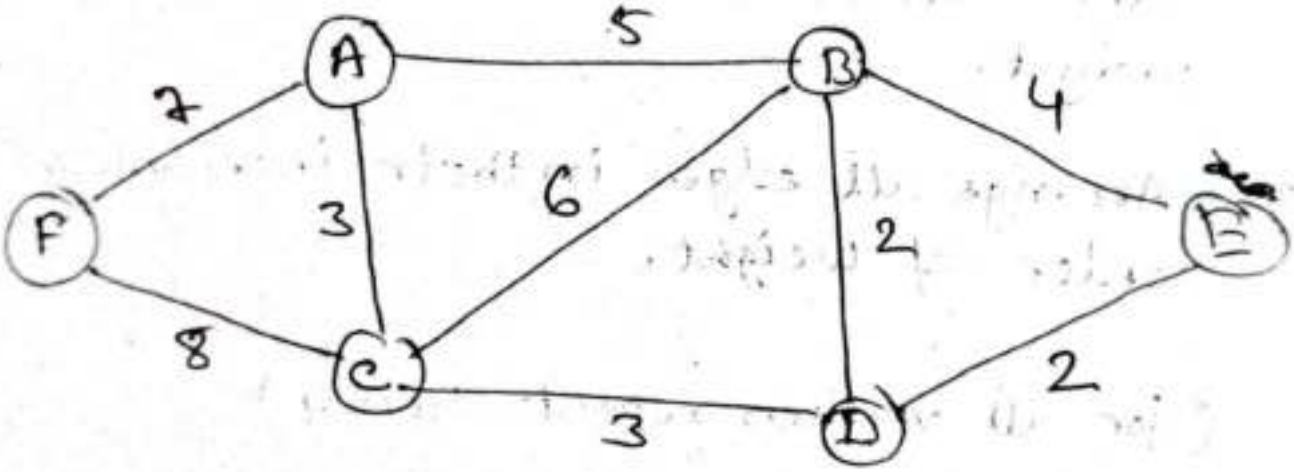
(for all vertices repeat step 4)

Step 4 :- Add the edge which has the least weight and with no cycle.

Exam



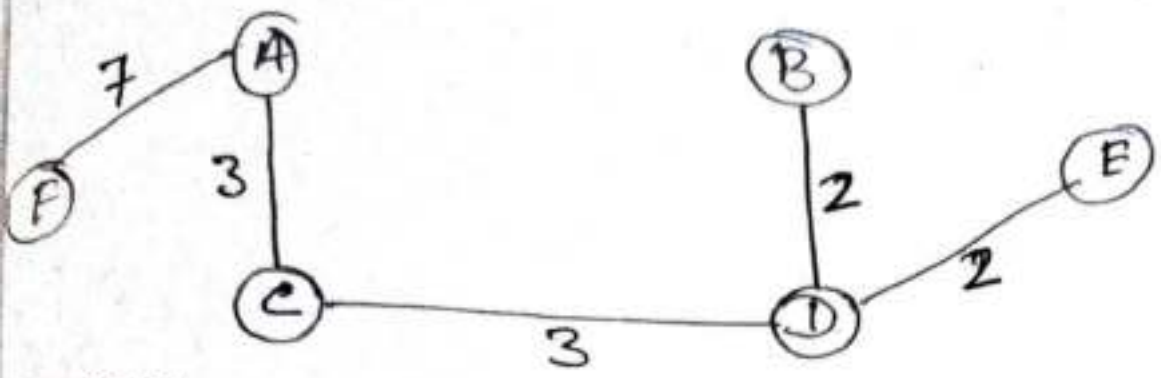

i) Remove loop & parallel edges.




write down all edges increasing order :-

BD = 2
DE = 2
AC = 3
CD = 3
BE = 4
AB = 5
BC = 6
AF = 7
FC = 8

* should not contain any cycle.




property

i) Does not contain any cycle.

ii) if the given graph contain n number of vertices, then MST contain same number of vertices (n) and n-1 number edges.

∴ there, $n = 6$ vertices

edges, $= n-1 = 6-1 = 5$

sum,

MST is = $2+2+3+3+7 = 17$.