Prim's

Minimum Spanning Tree

A spanning tree of a graph is a tree that has all the vertices of the graph connected by some edges.

A graph can have one or more number of spanning trees.

If the graph has **N vertices** then the spanning tree will have **N-1 edges**.

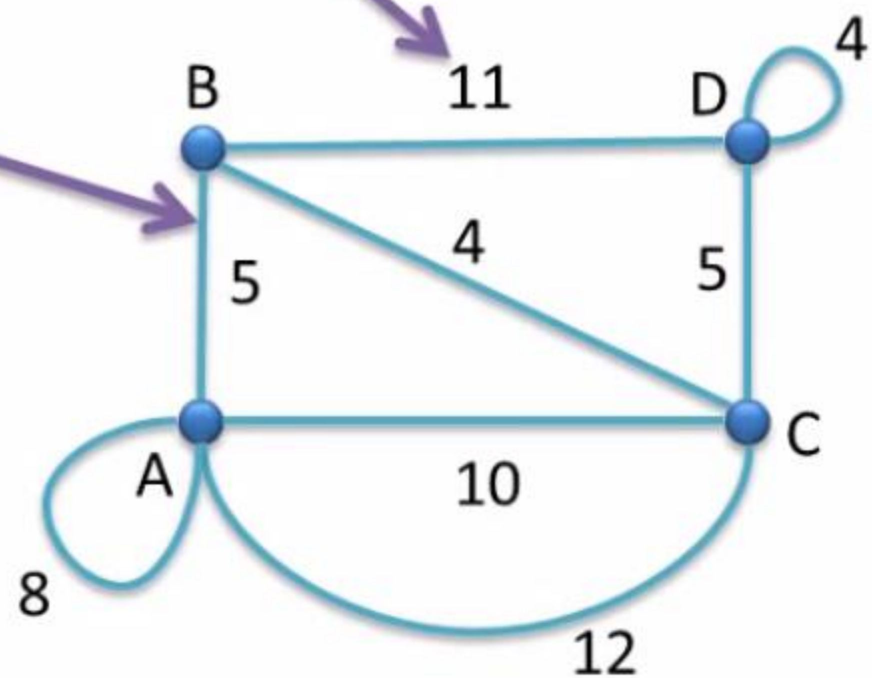
A minimum spanning tree (MST) is a spanning tree that has the minimum weight than all other spanning trees of the graph.

PRIMS ALGORITHM

Here is our graph

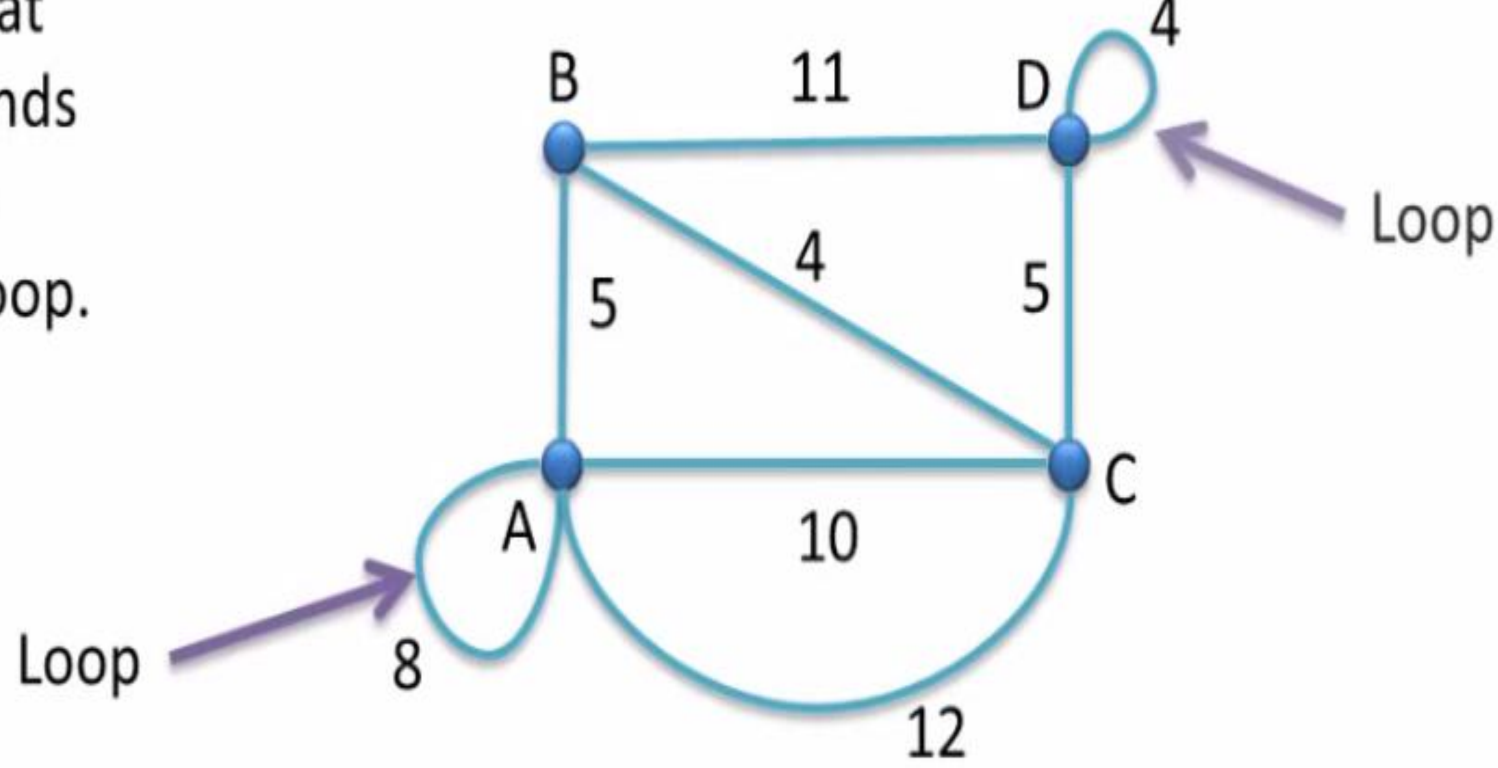
And this represents the weight of the edge

This represents an edge



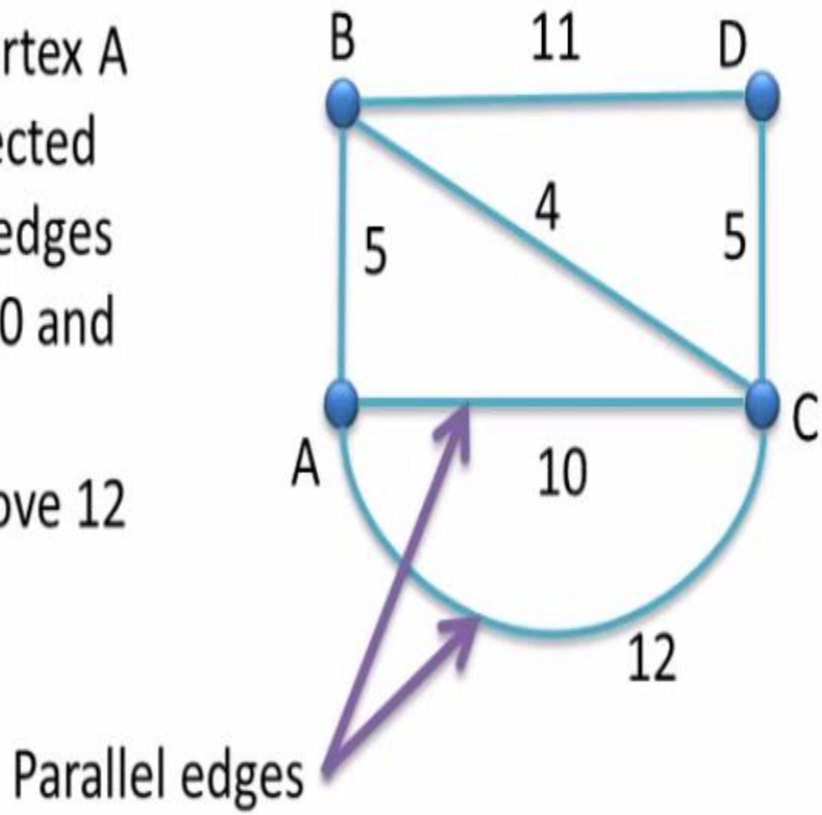
Step 1: Remove all the loops

Note!
Any edge that starts and ends at the same vertex is a loop.



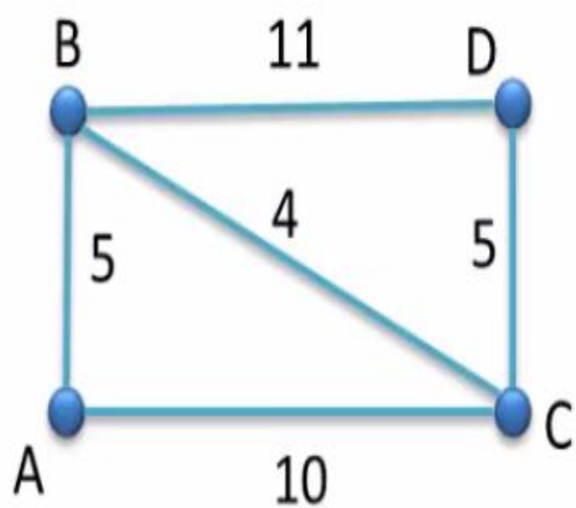
Step 2: Remove all parallel edges between two vertex except the one with least weight

Note!
In this graph, vertex A and C are connected by two parallel edges having weight 10 and 12 respectively.
So, we will remove 12 and keep 10.

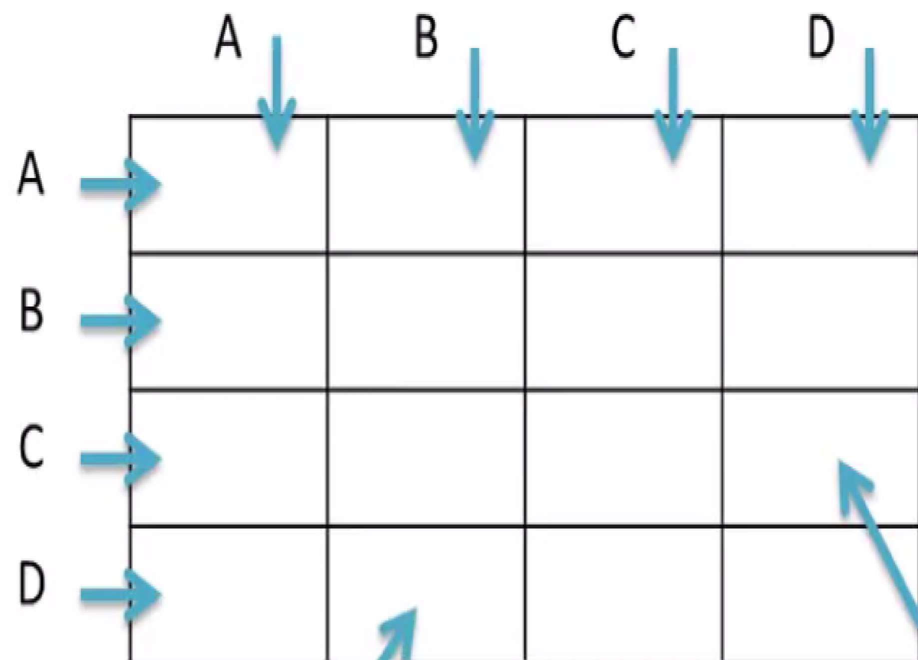


Step 3: Create table

As our graph has 4 vertices, so our table will have 4 rows and 4 columns



These are the columns

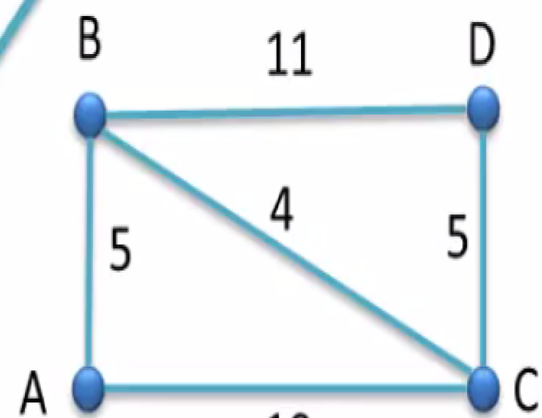


Note!
Row and Column name is same as the name of the vertex.

These are the rows

And this represent a cell CD. Where C is the Row name and D is the column name.

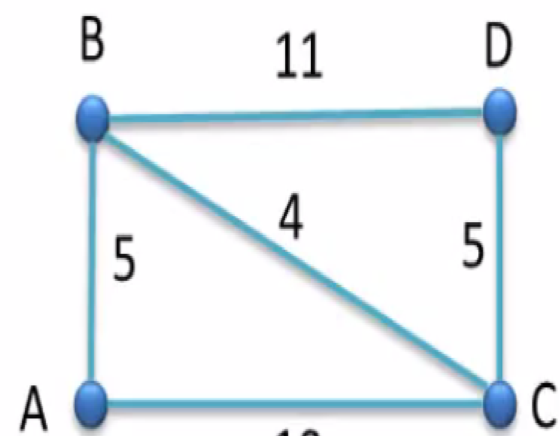
Similarly, this represent a cell DB. Where D is the Row name and B is the column name.



We will now fill the other cells.

	A	B	C	D
A	0			
B		0		
C			0	
D				0

Now, put 0 in cells having same row and column name.

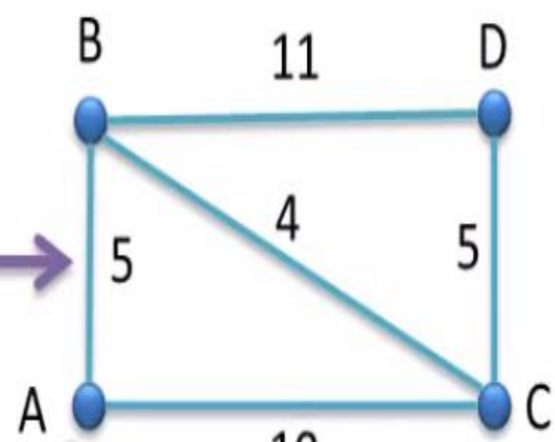


	A	B	C	D
A	0	5		
B	5	0		
C			0	
D				0

So we will write 5 in the cell AB and BA.

Find the edge that **directly** connects vertex A and B.

In this case, we have an edge of weight 5 that directly connect A and B.

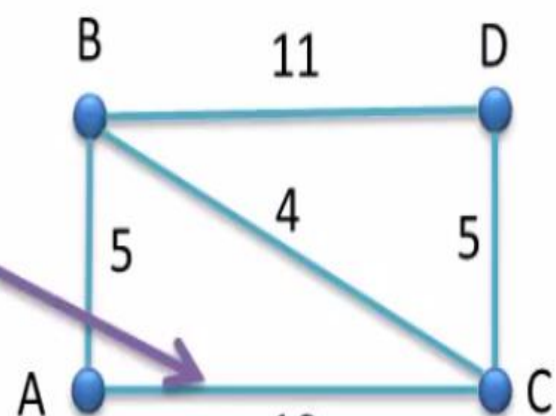


	A	B	C	D
A	0	5	10	
B	5	0		
C	10		0	
D				0

Find the edge that **directly** connects vertex A and C.

So we will write 10 in the cell AC and CA.

In this case, we have an edge of weight 10 that directly connect A and C.

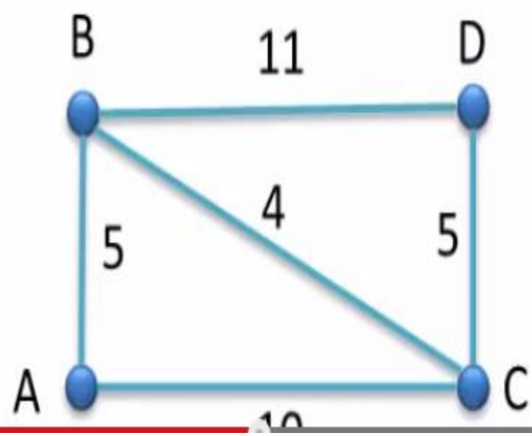


Find the edge that **directly** connects vertex A and D.

	A	B	C	D
A	0	5	10	∞
B	5	0		
C	10		0	
D	∞			0

So we will write ∞ in the cell AD and DA.

∞ denotes **Infinity**.



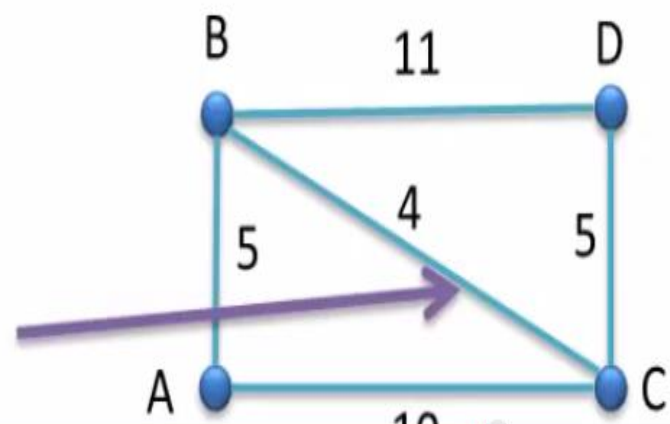
In this case, we don't have an edge that directly connects A and D

	A	B	C	D
A	0	5	10	∞
B	5	0	4	
C	10	4	0	
D	∞			0

So we will write 4 in the cell BC and CB.

Find the edge that **directly** connects vertex B and C.

In this case, we have an edge of weight 4 that directly connect B and C.

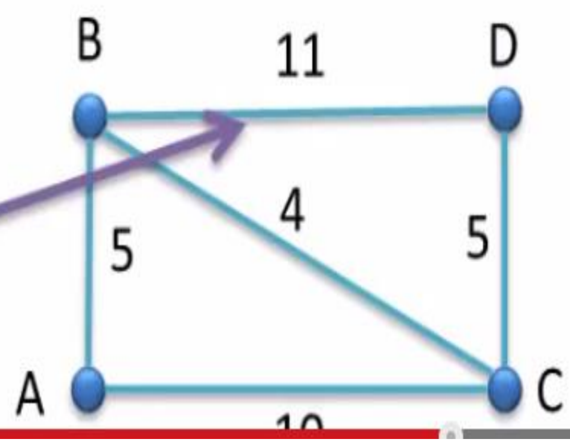


	A	B	C	D
A	0	5	10	∞
B	5	0	4	
C	10	4	0	
D	∞			0

Find the edge that **directly** connects vertex B and D.

So we will write 11 in the cell BD and DB.

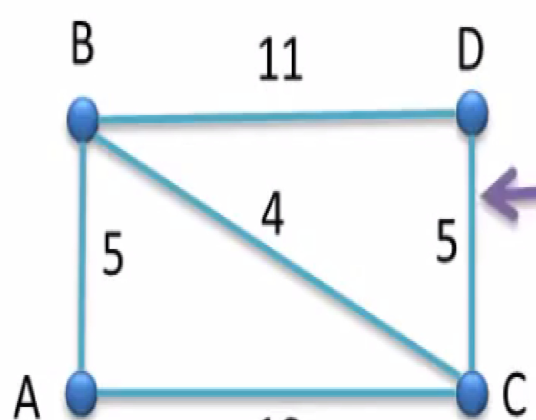
In this case, we have an edge of weight 11 that directly connect B and D.



	A	B	C	D
A	0	5	10	∞
B	5	0	4	11
C	10	4	0	5
D	∞	11	5	0

Find the edge that **directly** connects vertex C and D.

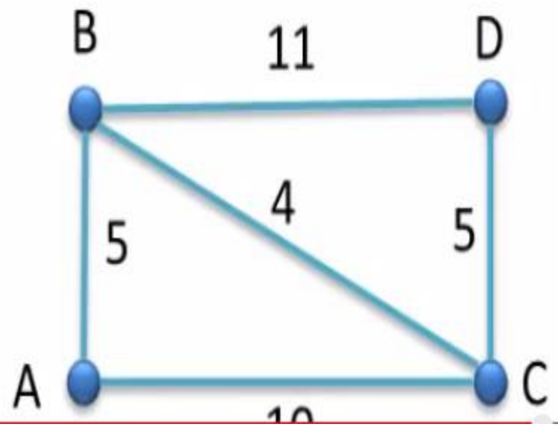
So we will write 5 in the cell CD and DC.



In this case, we have an edge of weight 5 that directly connect C and D.

	A	B	C	D
A	0	5	10	∞
B	5	0	4	11
C	10	4	0	5
D	∞	11	5	0

Our table is completely filled, so our next job is to find the MST.



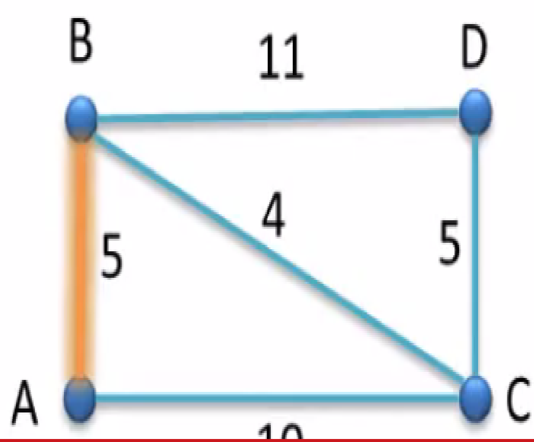
Start from vertex A.

Find the smallest value in the A-row.

Note!
We will not consider 0 as it will correspond to the same vertex.

	A	B	C	D
A	0	5	10	∞
B	5	0	4	11
C	10	4	0	5
D	∞	11	5	0

Smallest value in cell AB



5 is the smallest unmarked value in the A-row.

So, we will mark the edge connecting vertex A and B

Tick 5 in AB and BA cell.

As we connected vertex A and B in the previous step, so we will now find the smallest value in the A-row and B-row.

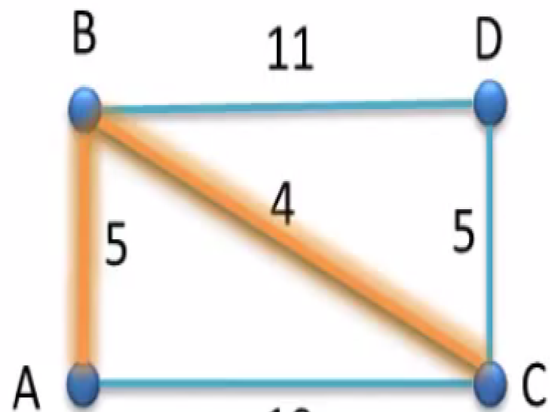
	A	B	C	D
A	0	5	10	∞
B	5	0	4	11
C	10	4	0	5
D	∞	11	5	0

Smallest value in cell BC

4 is the smallest unmarked value in the A-row and B-row.

So, we will mark the edge connecting vertex B and C

Note!
We will not consider 0 as it will correspond to the same vertex.



As vertex A-B and B-C were connected in the previous steps, so we will now find the smallest value in A-row, B-row and C-row.

	A	B	C	D
A	0	5 ✓	10	∞
B	5 ✓	0	4 ✓	11
C	10	4 ✓	0	5 ✓
D	∞	11	5 ✓	0

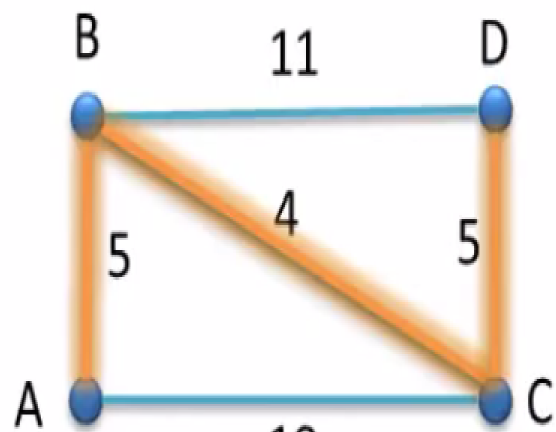
Smallest value in cell CD

5 is the smallest unmarked value in the A-row, B-row and C-row.

So, we will mark the edge connecting vertex C and D

Tick 5 in CD and DC cell.

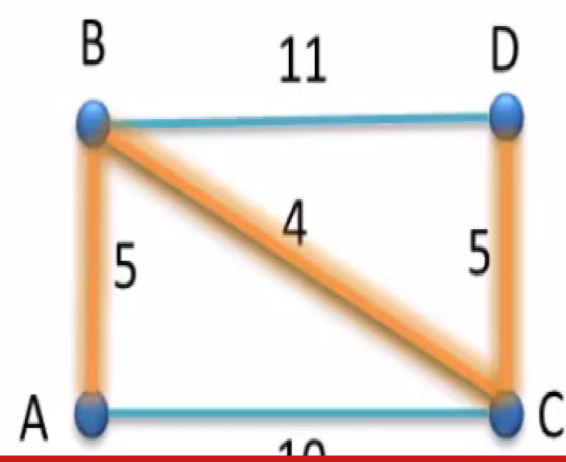
Note!
We will not consider 0 as it will correspond to the same vertex.



	A	B	C	D
A	0	5✓	10	∞
B	5✓	0	4✓	11
C	10	4✓	0	5✓
D	∞	11	5✓	0

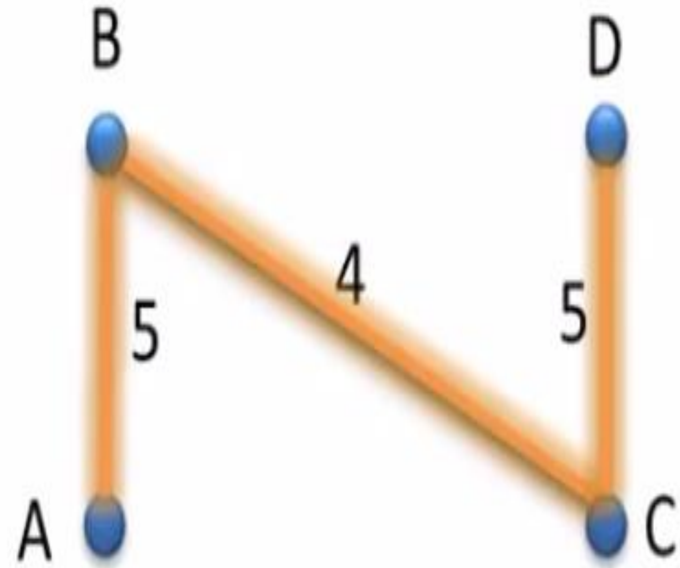
Note!
A spanning tree with 4 vertices will have 3 edges.

As we have marked all the 4 vertices, so we will stop here.

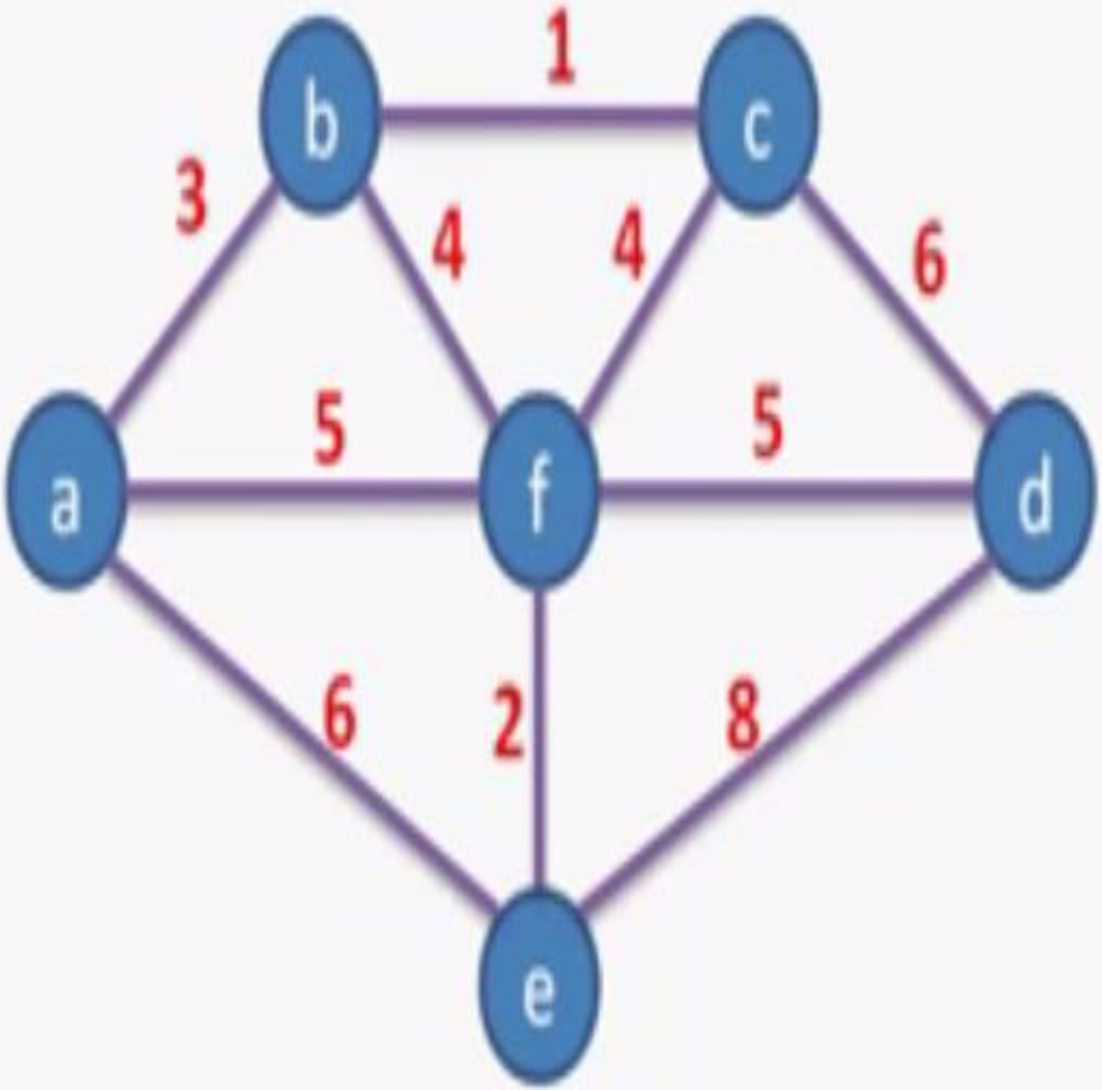


Our required Minimum Spanning Tree (MST) is

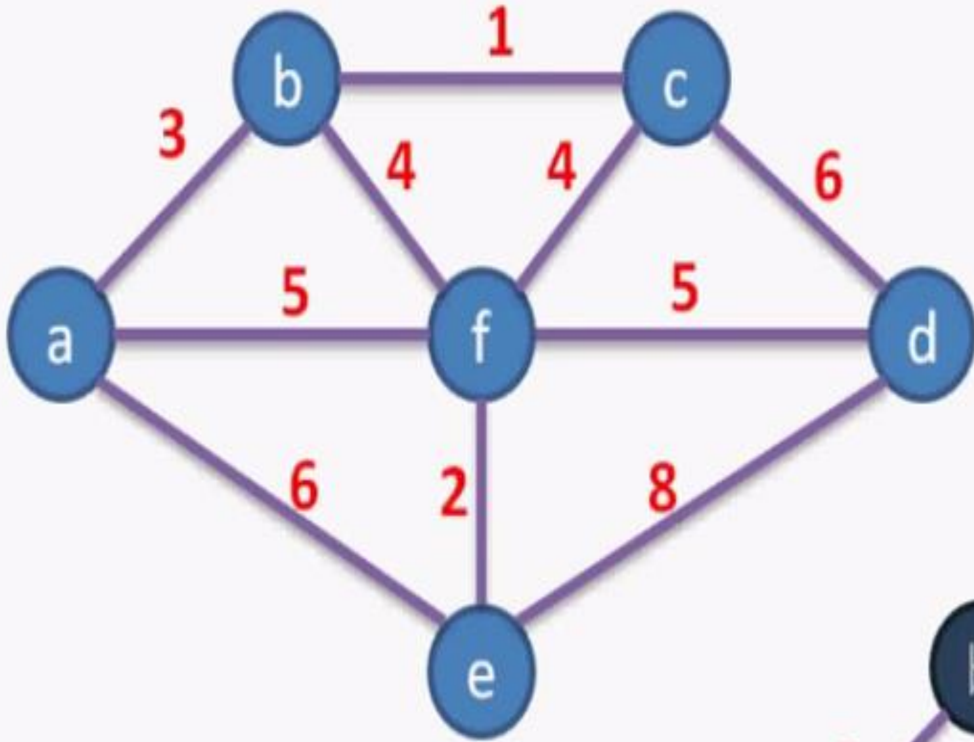
Weight of the MST
= 5+4+5
= 14 unit

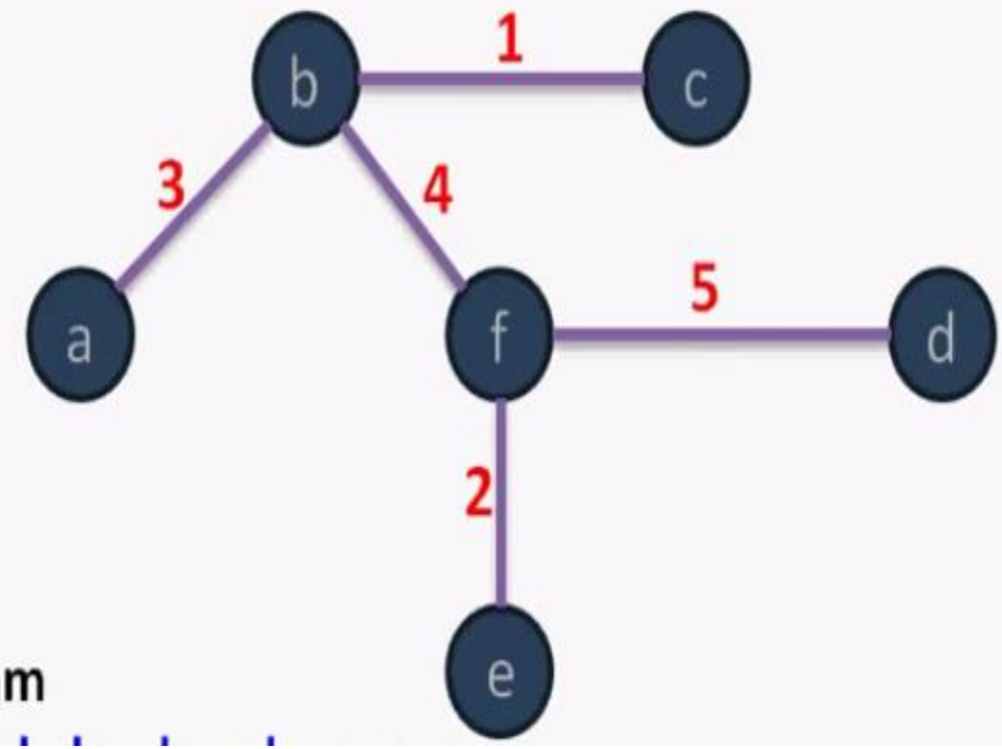


Prims Algorithm Example



Prims Algorithm
Example





Prims Algorithm

Vt → Represents the vertices that are already Visited.

Et → Represents all the edges that are already visited.

V belongs to set of visited vertex.
U belongs to set of unvisited vertex.

Add new vertex {U} to visited vertex Vt

Add new edge e to visited edge.

Algorithm Prim(G)

Vt ← {v0} //Set of visited vertices

Et ← ϕ

for i ← 1 to |V| -1 do

find minimum edge e between vertices v and u such that v is in Vt and u is in V - Vt

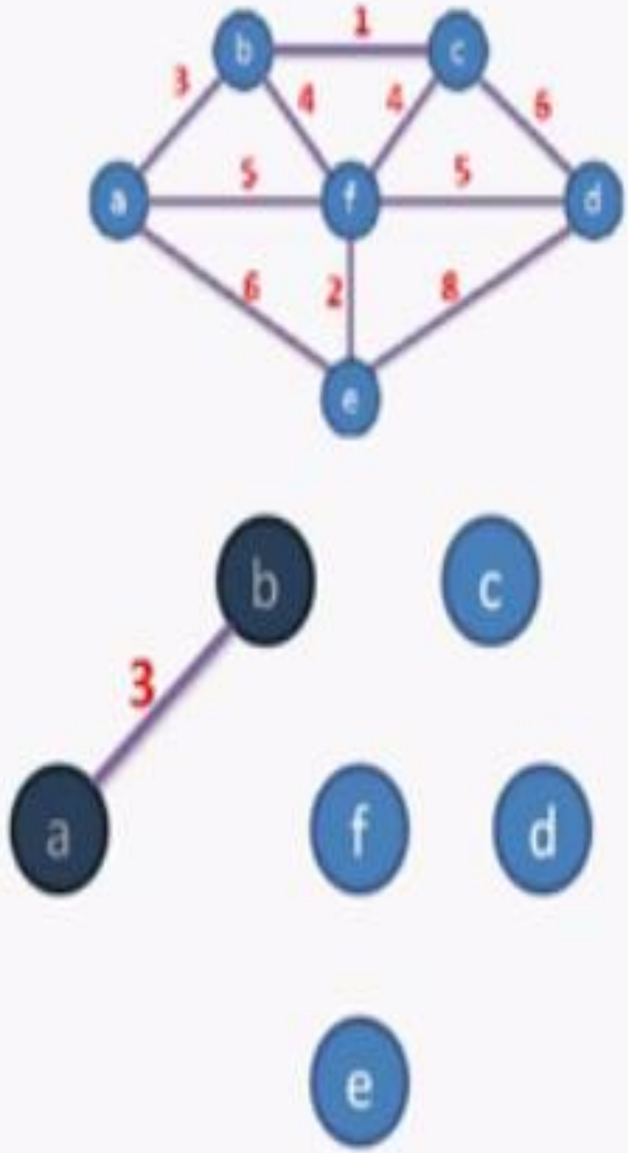
//Add u to Vt

Vt ← Vt U {u}

//Add the edge to the spanning tree

Et ← Et U {e}

Prims Algorithm



Algorithm Prim(G)

Vt ← {v0} //Set of visited vertices

Et ← ϕ

for i ← 1 to |V| -1 do

find minimum edge e between vertices v and u such that v is in Vt and u is in V - Vt

//Add u to Vt

Vt ← Vt U {u}

//Add the edge to the spanning tree

Et ← Et U {e}

Prims Algorithm

ANALYSIS

- ***Prim's Algorithm produces a minimum spanning tree of G.***

Proof.

- For Prim's Algorithm, it is also very easy to show that it only adds edges belonging to every minimum spanning tree.
- Indeed, in each iteration of the algorithm, there is a set $S \subseteq V$ on which a partial spanning tree has been constructed, and a node v and edge e are added that minimize the quantity
- $\min_{e=(u,v):u \in S} c_e$.
- By definition, e is the cheapest edge with one end in S and the other end in $V - S$, and so by the Cut Property (4.17) it is in every minimum spanning tree.

- It is also straightforward to show that Prim's Algorithm produces a spanning tree of G, and hence it produces a minimum spanning tree.