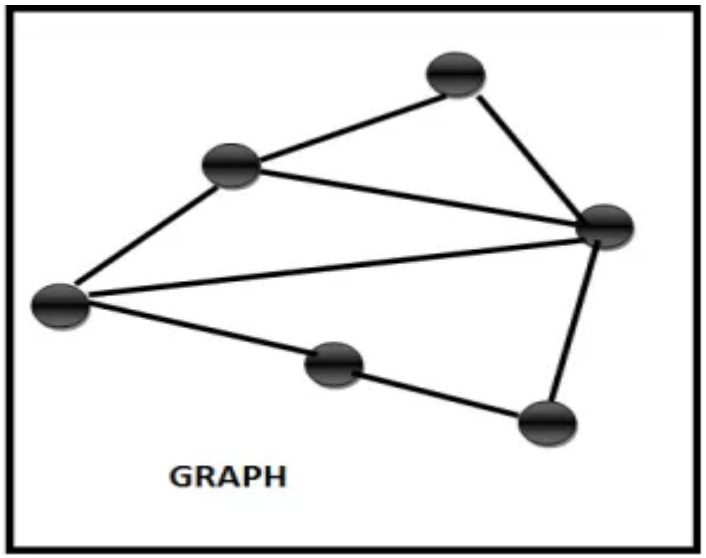
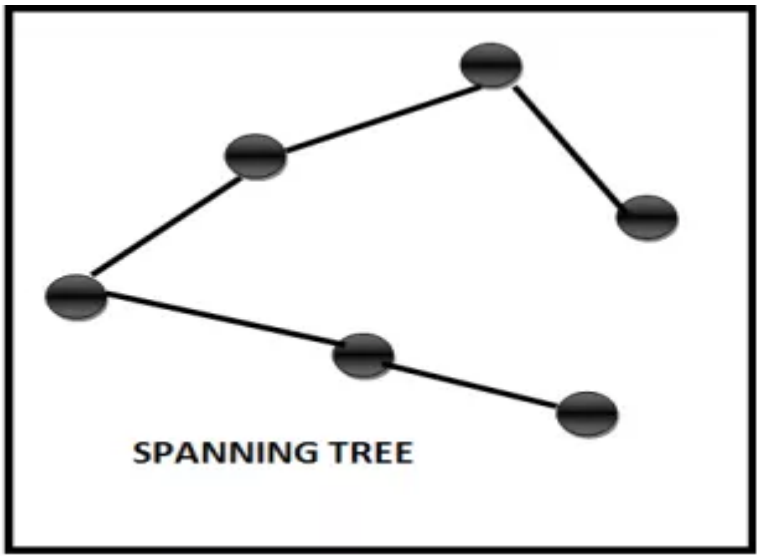
MINIMUM SPANNING TREE PROBLEM

SPANNING TREE

- A spanning tree of an undirected connected graph is its **connected acyclic sub graph** (i.e., a tree) that contains all the vertices of the graph.



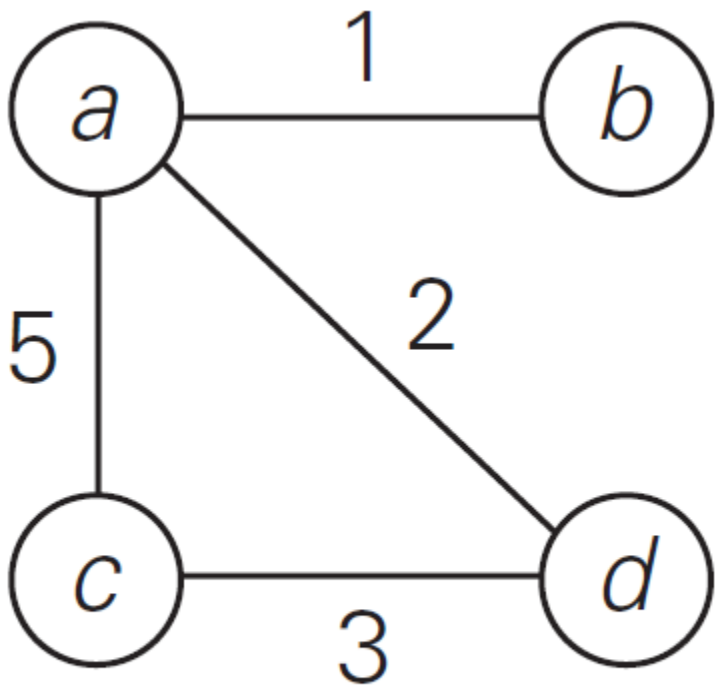


MINIMUM SPANNING TREE

- If such a graph has weights assigned to its edges, a minimum spanning tree is its spanning tree of the smallest weight, where the weight of a tree is defined as the sum of the weights on all its edges

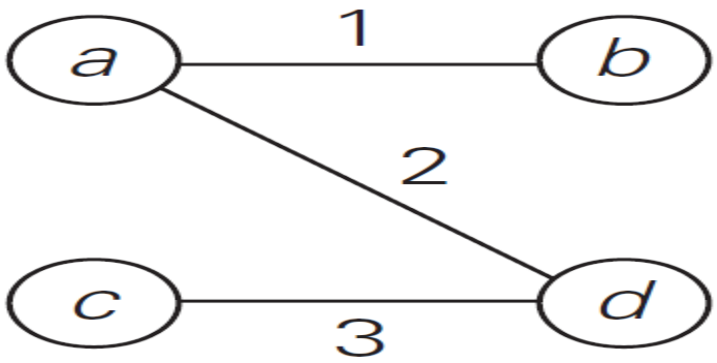
- The minimum spanning tree problem is the problem of finding a minimum spanning tree for a given

MINIMUM SPANNING TREE

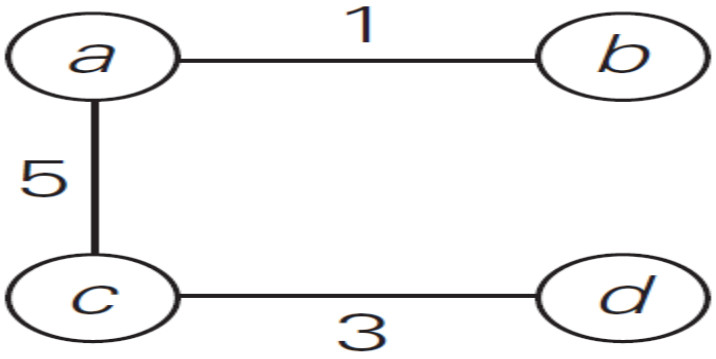


graph

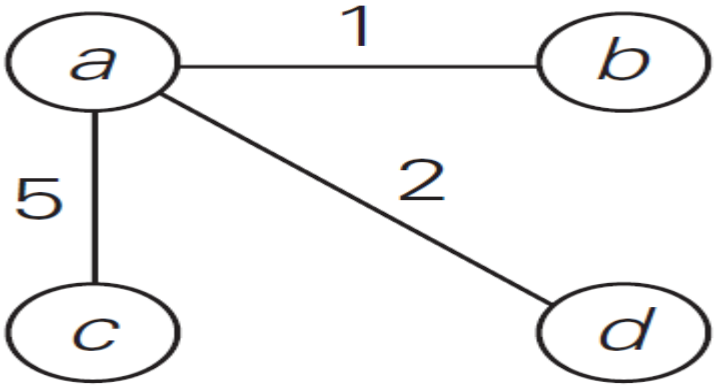
MINIMUM SPANNING TREE



$w(T_1) = 6$



$w(T_2) = 9$



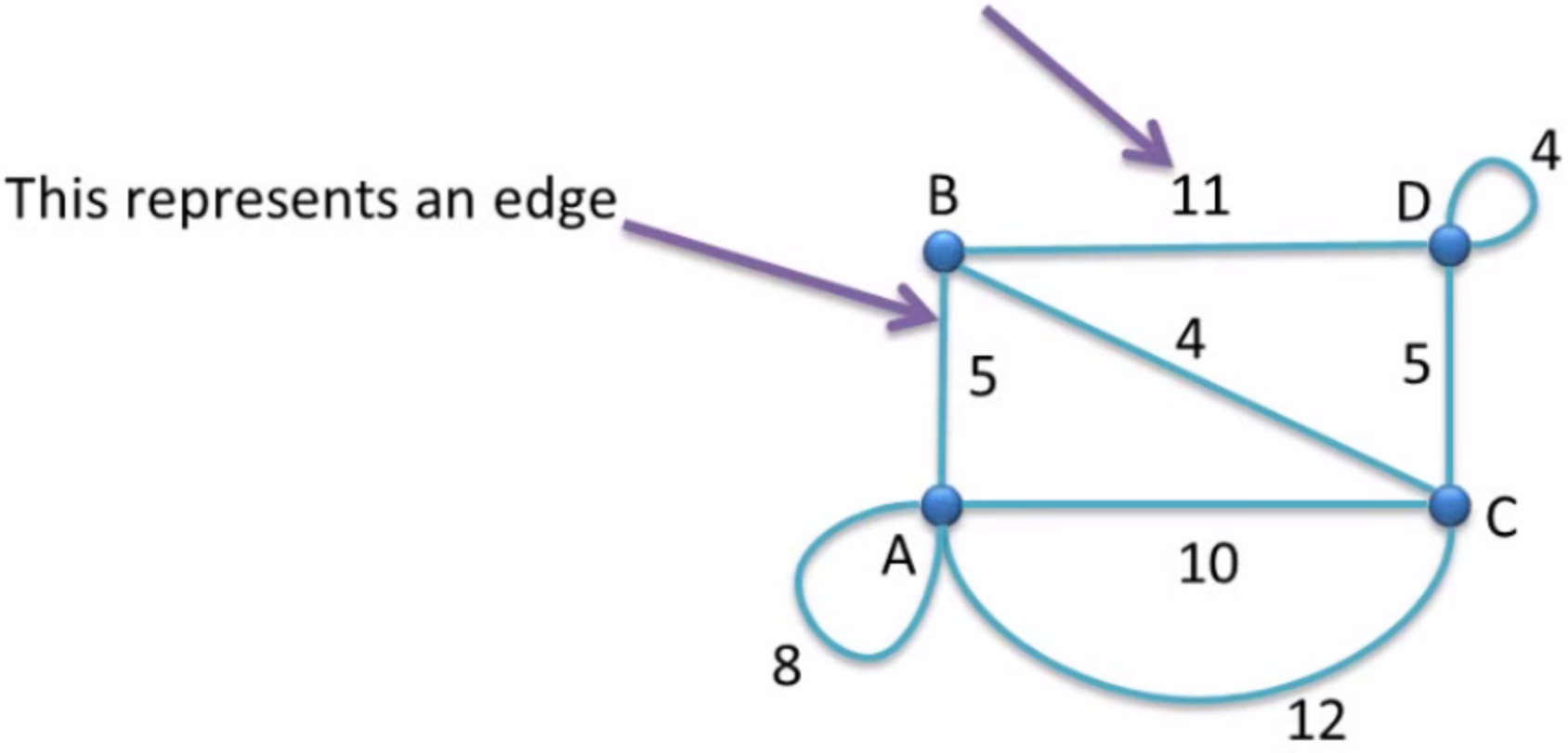
$w(T_3) = 8$

PRIM'S ALGORITHM

Here is our graph

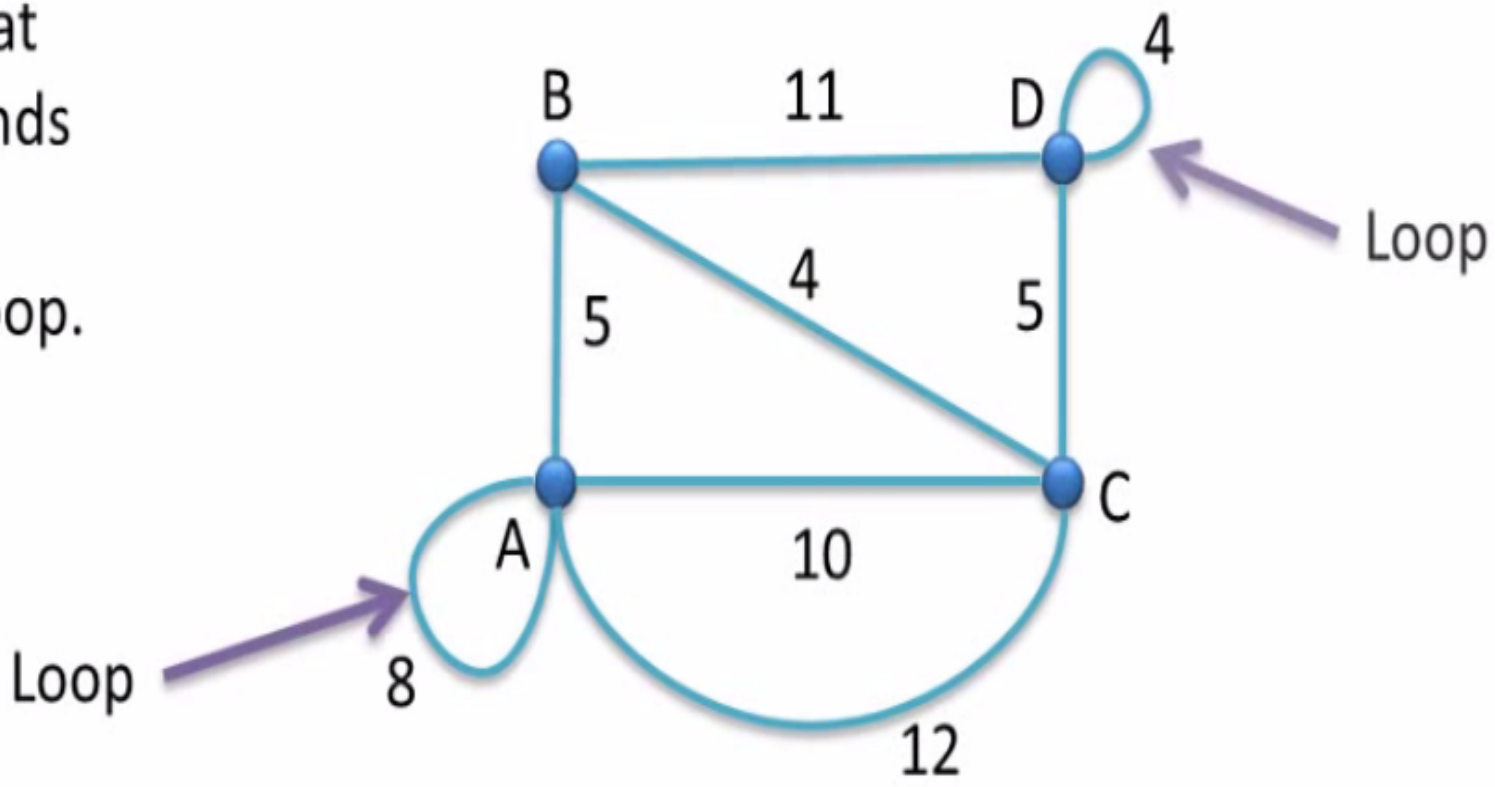
And this represents the weight of the edge

This represents an edge



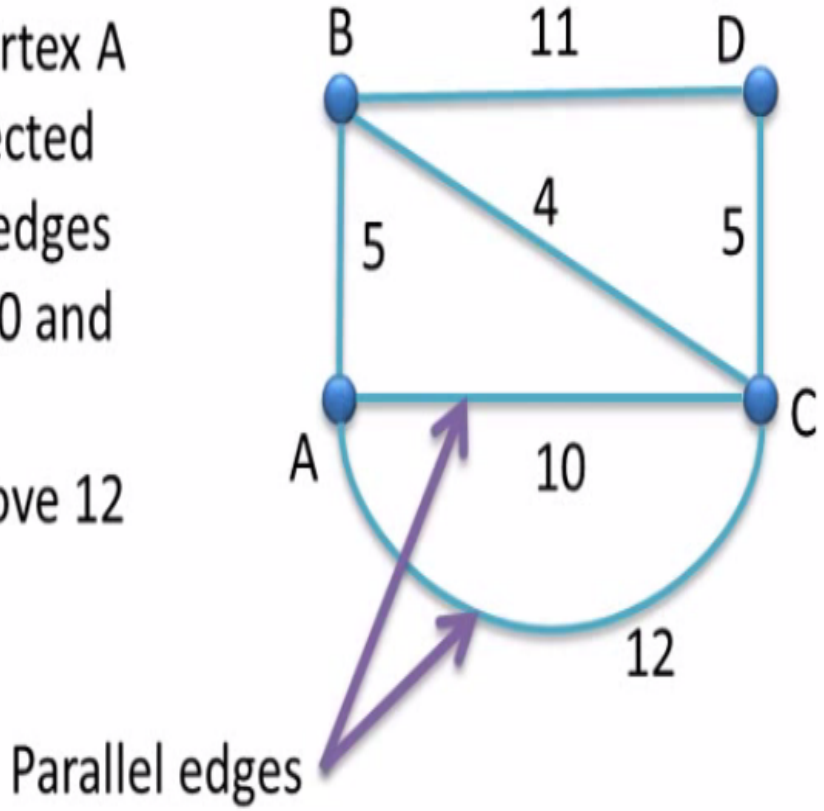
Step 1: Remove all the loops

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



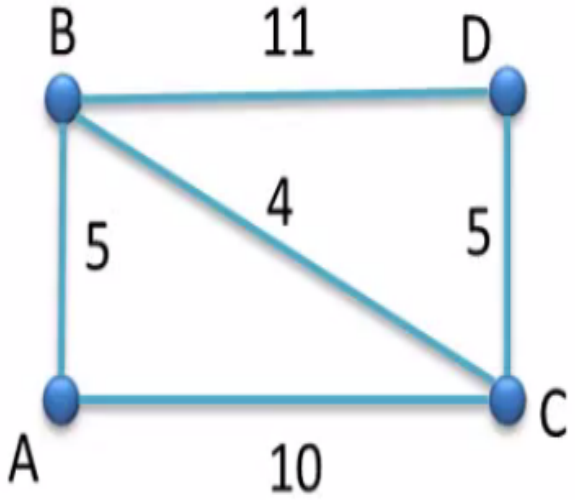
Step 2: Remove all parallel edges between two vertex except the one with least weight

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.

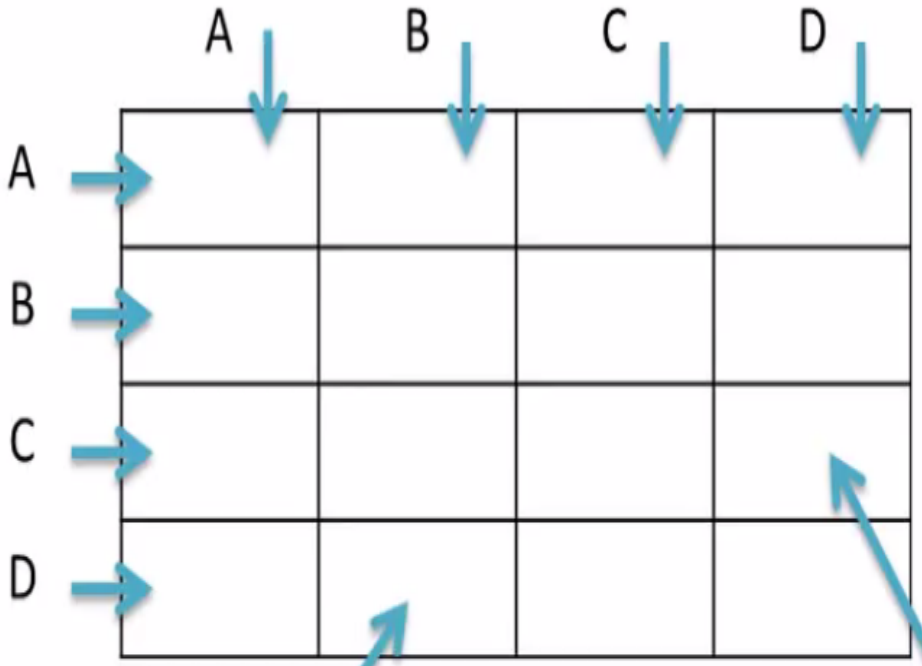


Step 3: Create table

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



These are the columns

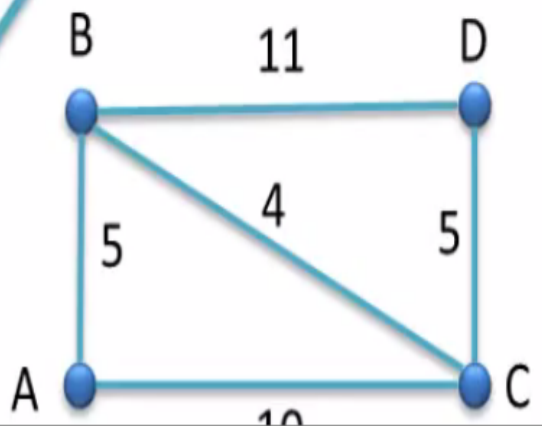


These are the rows

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

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

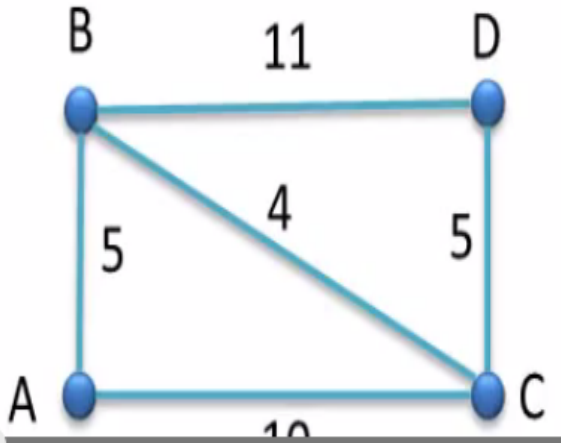
Similarly, this represent a cell DB. Where D is the Row name and B is the column name.



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.

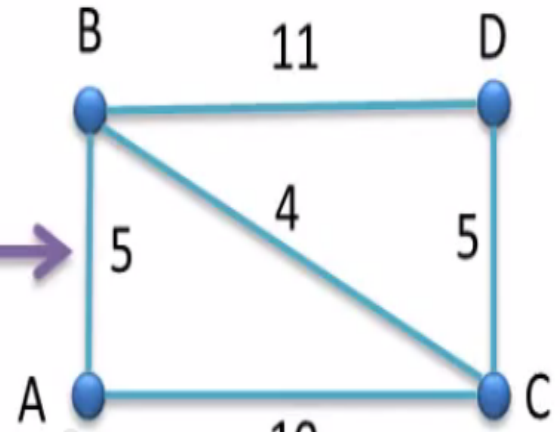


	A	B	C	D
A	0	5		
B	5	0		
C			0	
D				0

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.

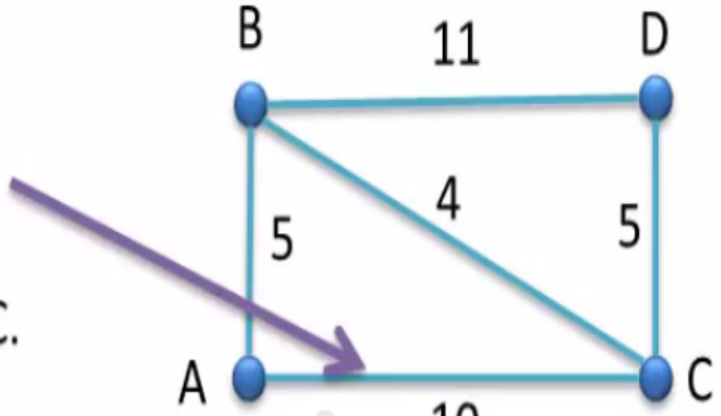


So we will write 10 in the cell AC and CA.

	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.

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



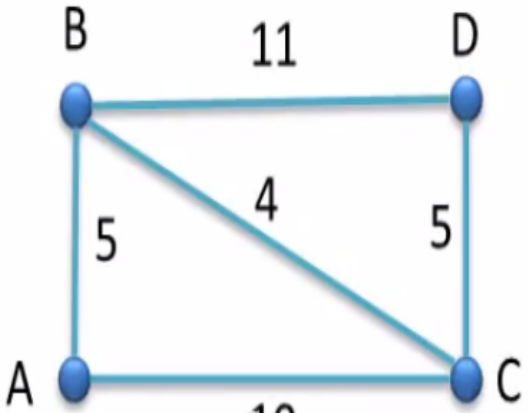
	A	B	C	D
A	0	5	10	∞
B	5	0		
C	10		0	
D	∞			0

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

∞ denotes **Infinity**.

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

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

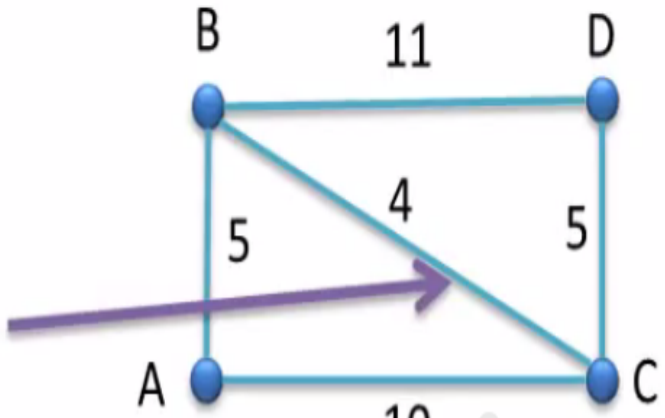


	A	B	C	D
A	0	5	10	∞
B	5	0	4	
C	10	4	0	
D	∞			0

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.

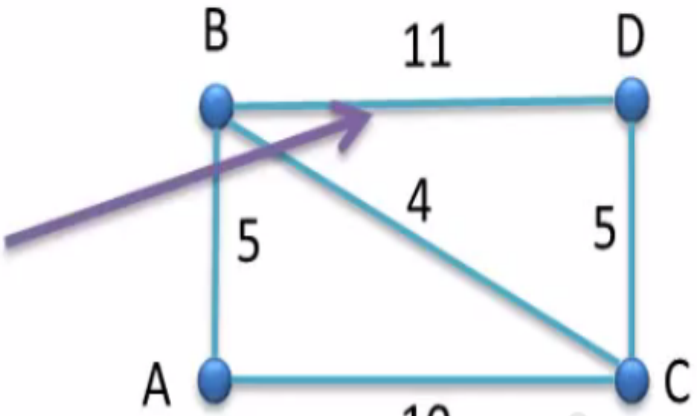


	A	B	C	D
A	0	5	10	∞
B	5	0	4	
C	10	4	0	
D	∞			0

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

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

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

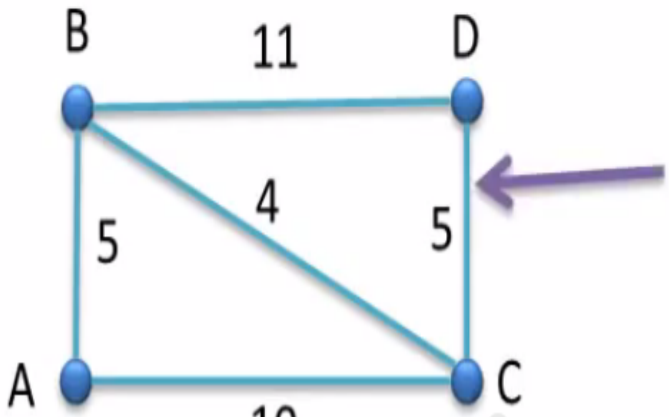


Next

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

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

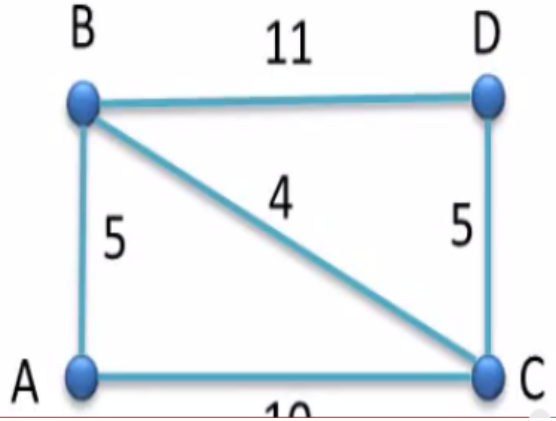
Find the edge that **directly** connects vertex C and D.



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.



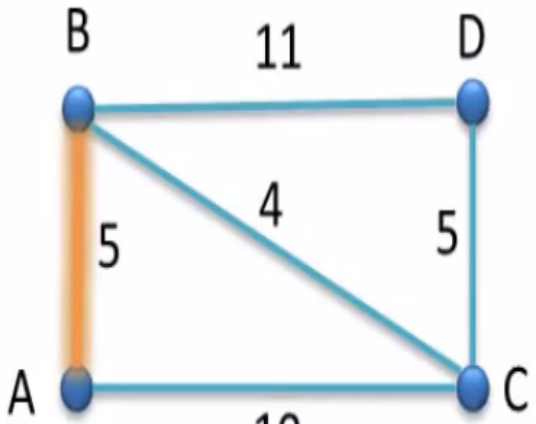
Start from vertex A.

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.

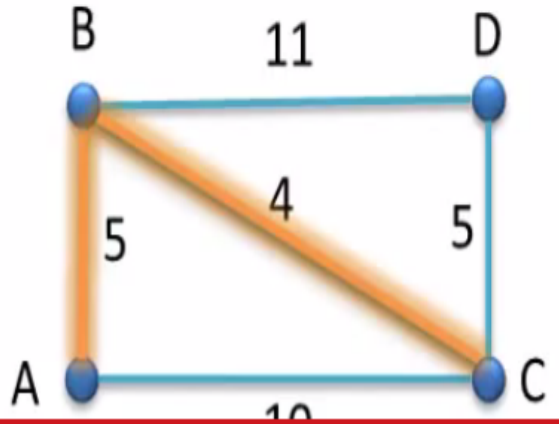
	A	B	C	D
A	0	5✓	10	∞
B	5✓	0	4	11
C	10	4	0	5
D	∞	11	5	0

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

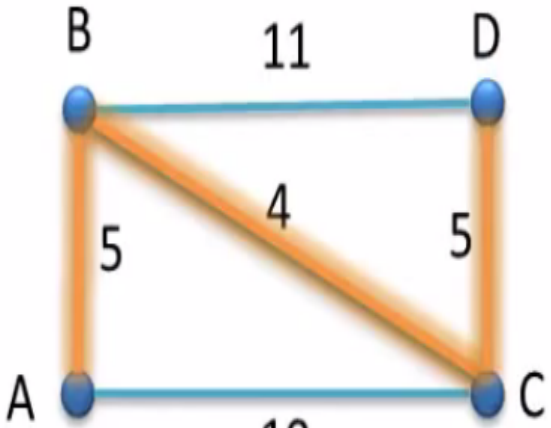
Smallest value in cell CD

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.

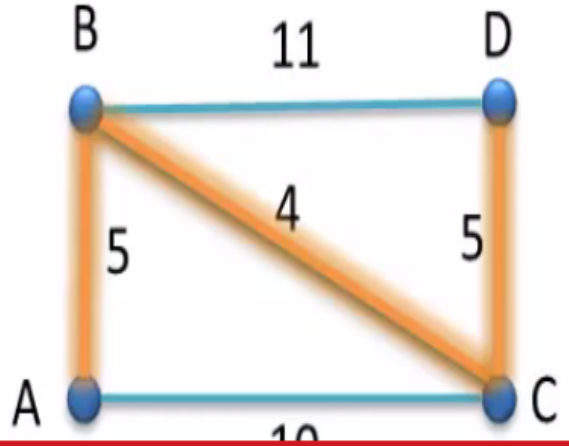
Note!
We will not consider 0 as it will correspond to the same vertex.



	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.

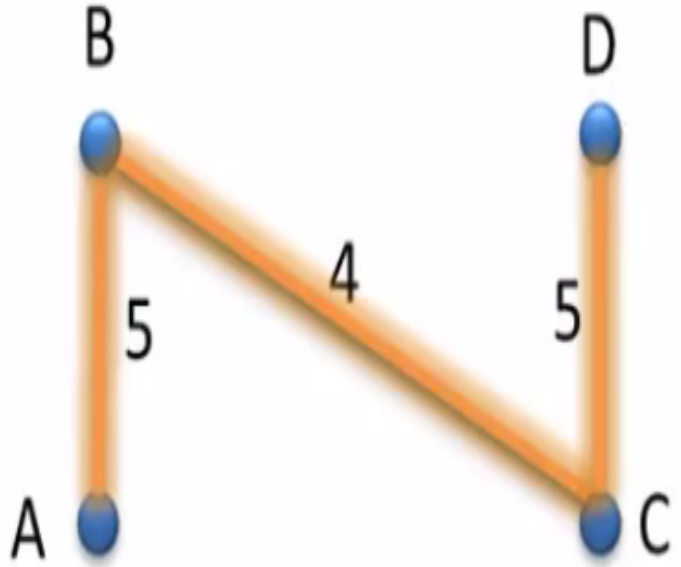


Our required Minimum Spanning Tree (MST) is

Weight of the MST

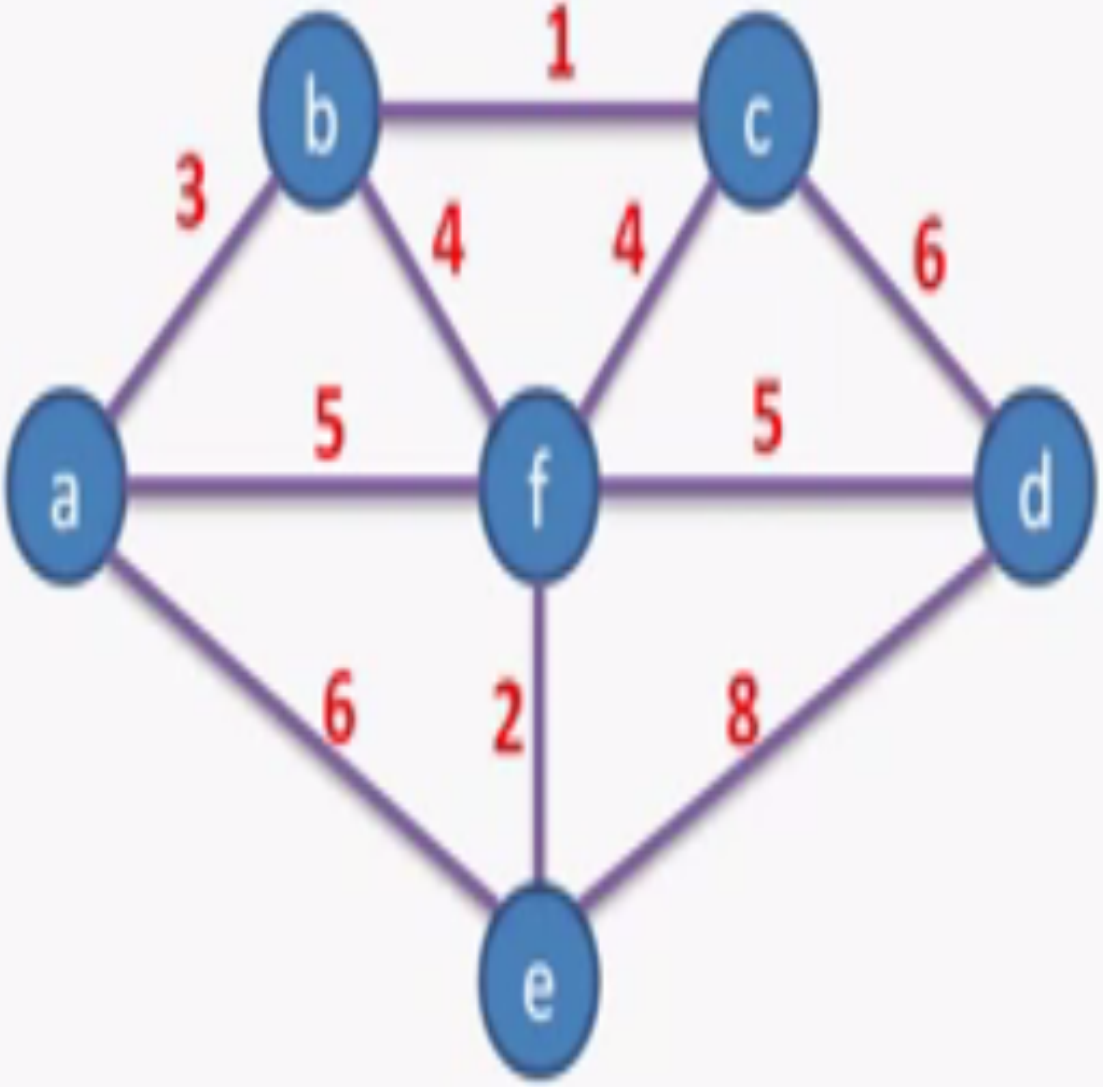
= 5+4+5

= 14 unit

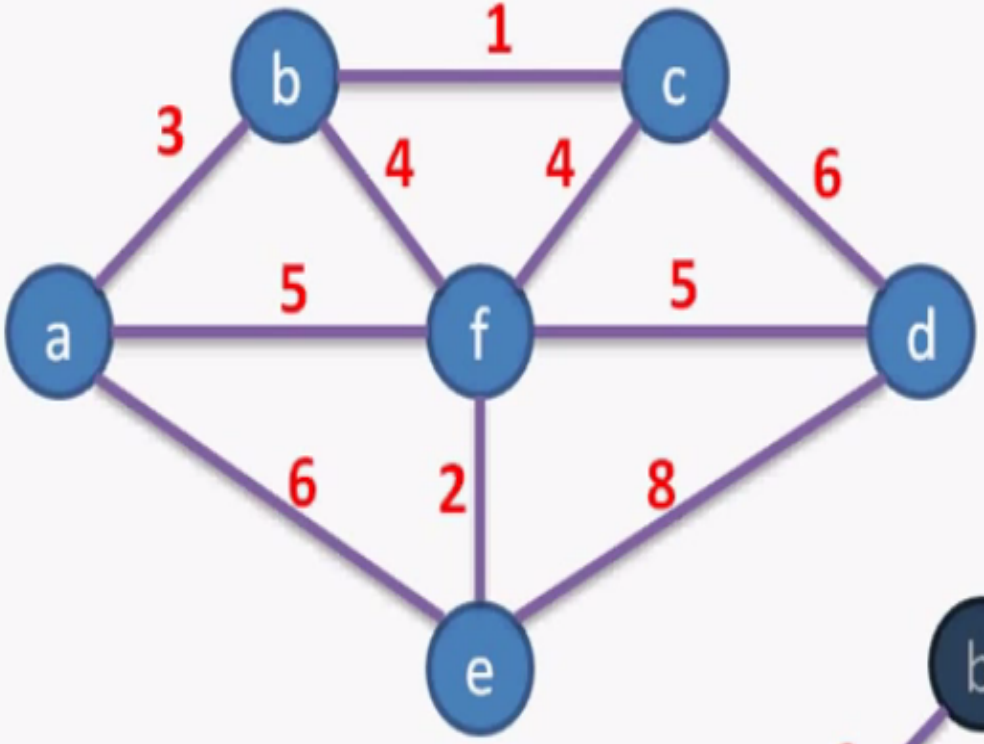


PRIM'S ALGORITHM

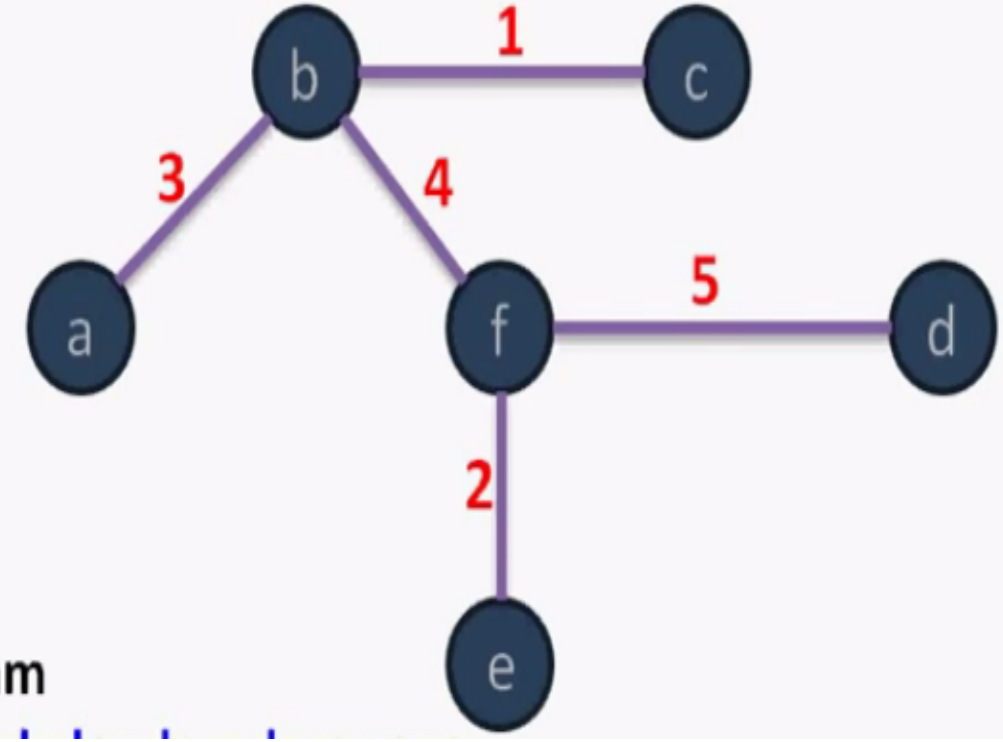
Prim's Algorithm (G, c)
Initially $S = \{s\}$ and $T = \emptyset$
While $S \neq V$
    Select a node $v \notin S$ which has an edge into S and
        for which the attachment cost $\min_{e=(u,v):u \in S} c_e$ is
            as small as possible
    Add v to S
    Add the edge e where the minimum is obtained to T
EndWhile
Return the spanning tree T



Prims Algorithm Example



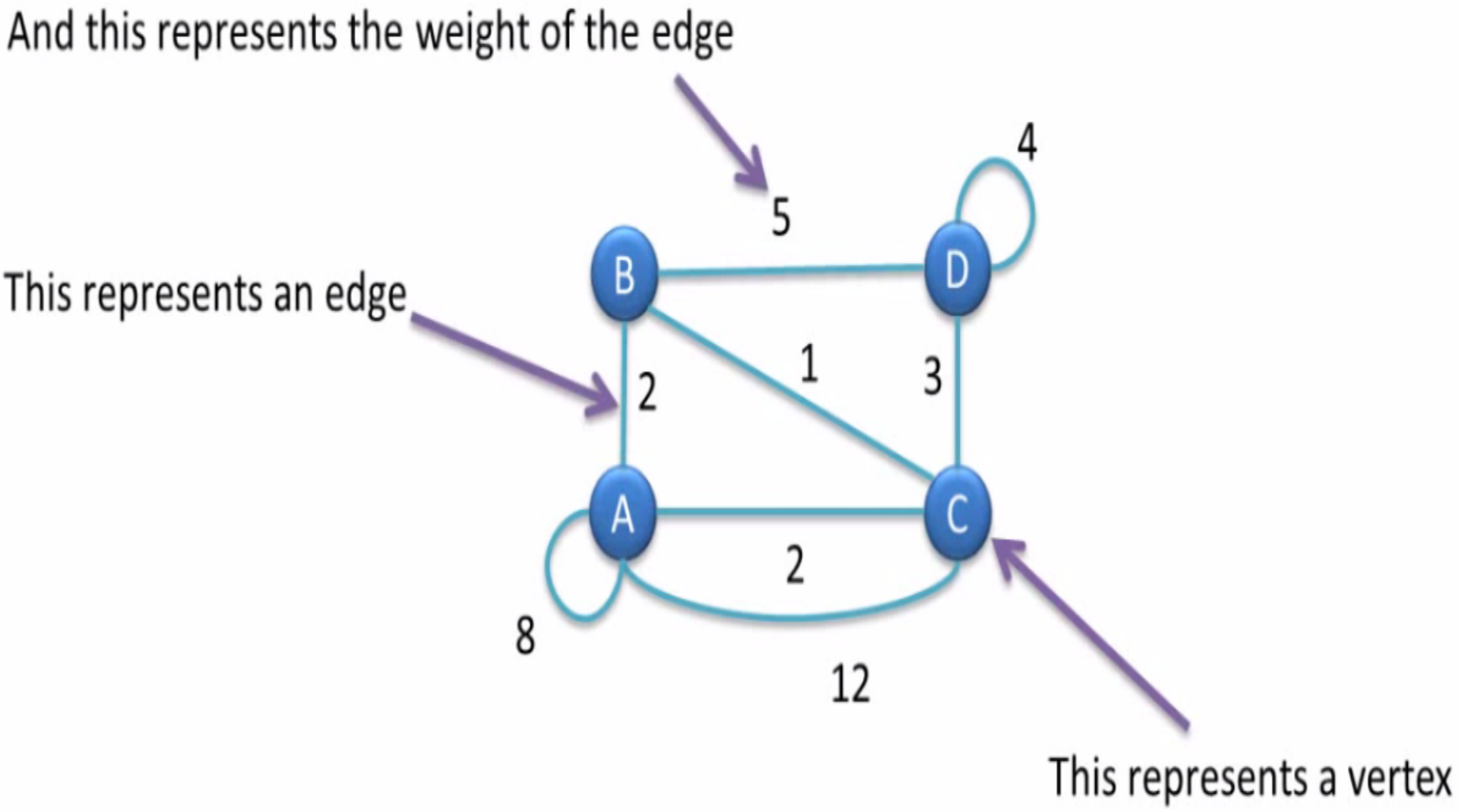
Prims Algorithm
Example



Prims Algorithm

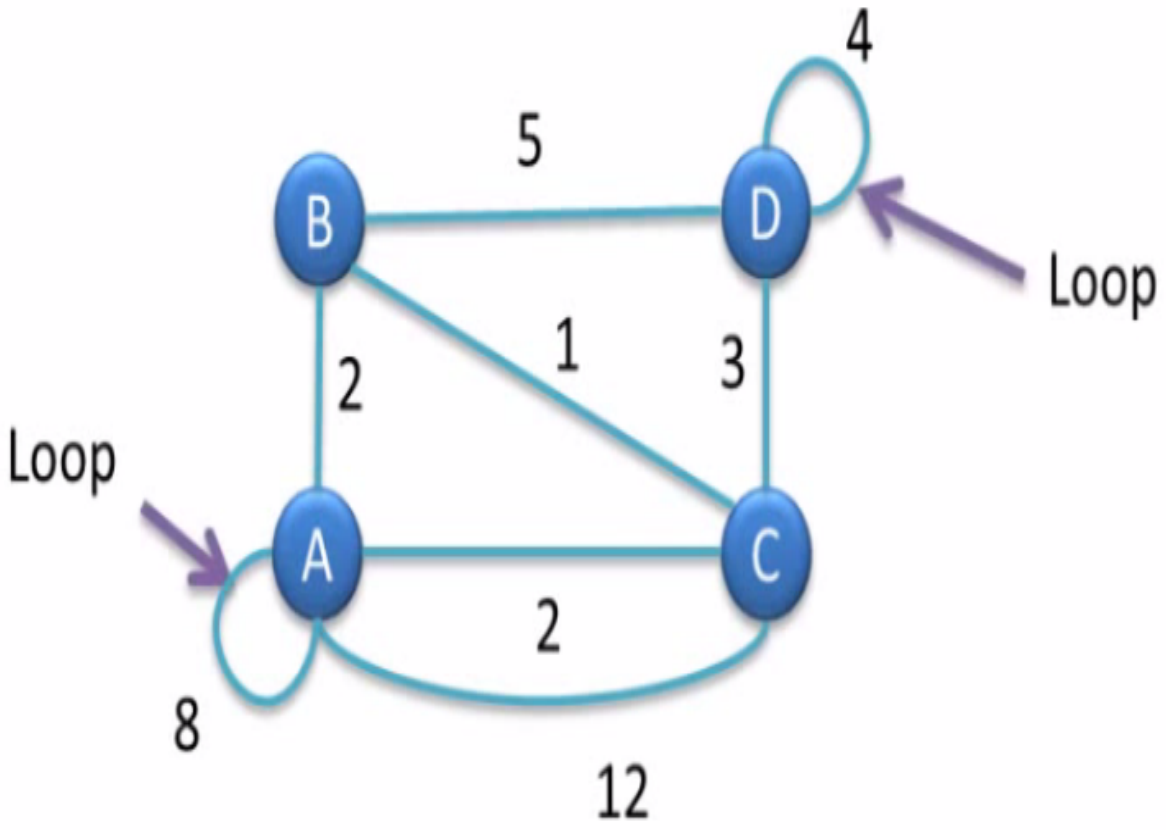
KRUSKAL'S ALGORITHM

Here is our graph

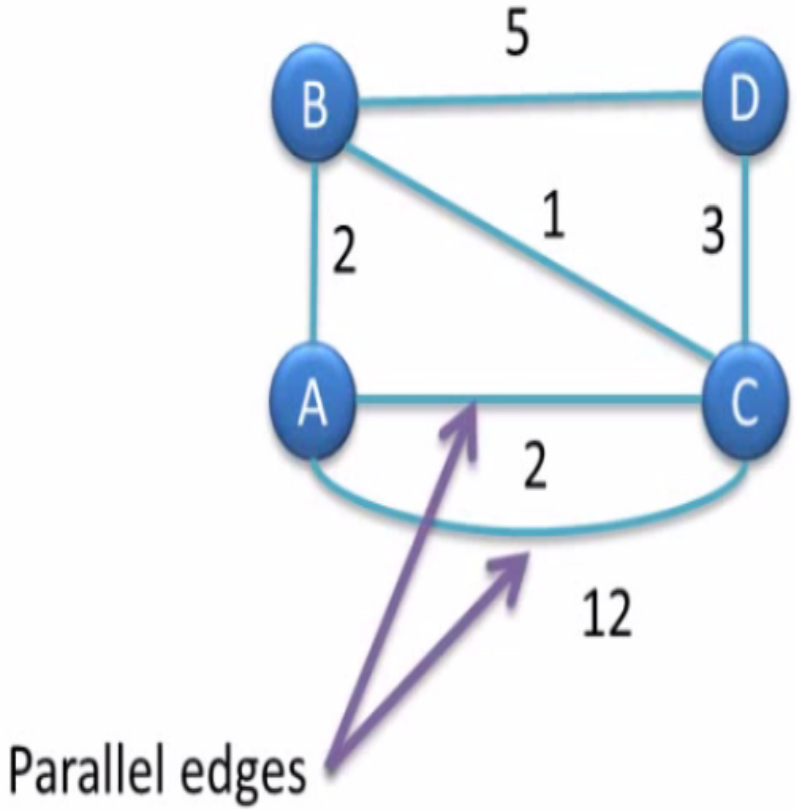


Step 1: Remove all the loops

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

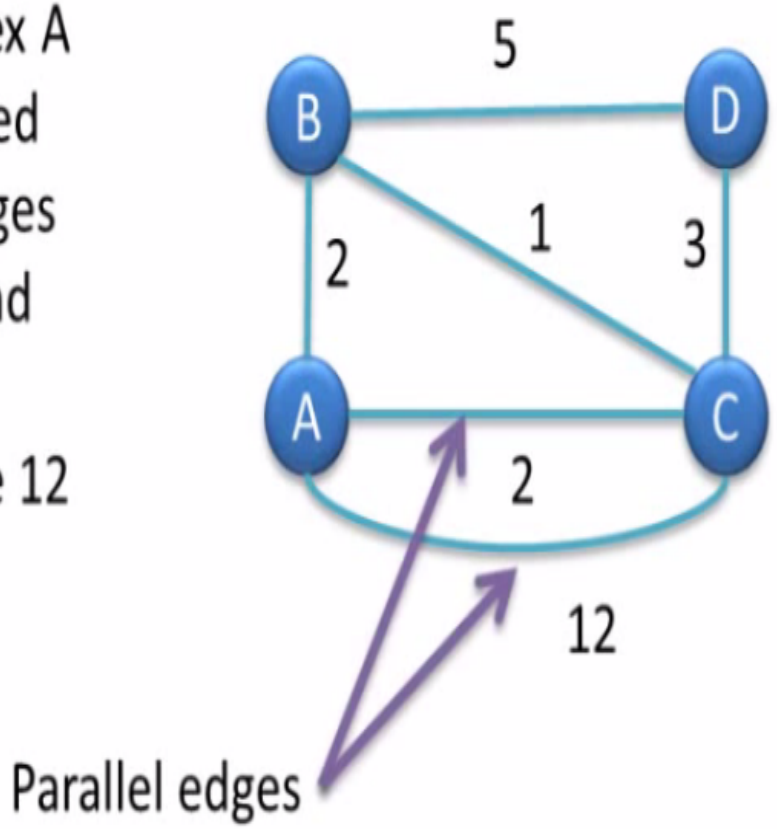


Step 2: Remove all parallel edges between two vertex except the one with least weight



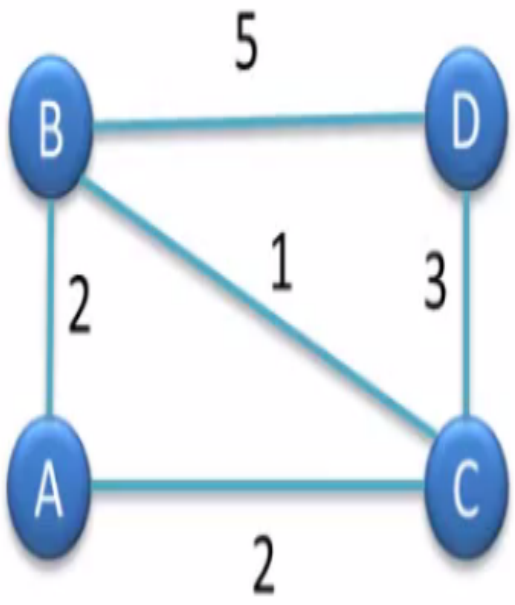
Step 2: Remove all parallel edges between two vertex except the one with least weight

Note!
In this graph, vertex A and C are connected by two parallel edges having weight 2 and 12 respectively.
So, we will remove 12 and keep 2.



Step 3: Create the edge table

An edge table will have name of all the edges along with their weight is ascending order.



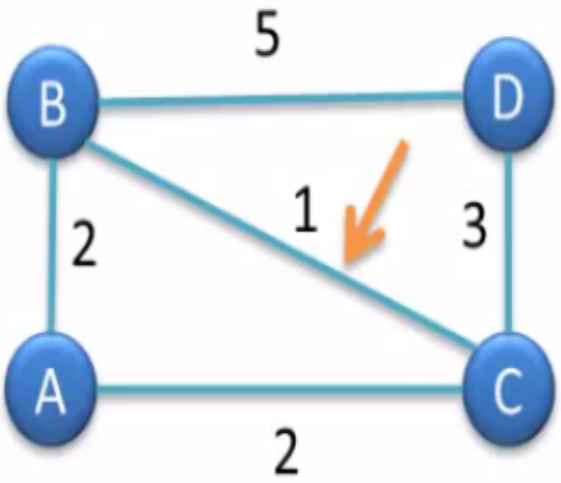
Now if you look at the graph, then you will notice that there are total 5 edges.

So our edge table will have 5 columns.

Edge					
Weight					

Now look at the graph and fill the 1st column with the edge of minimum weight.

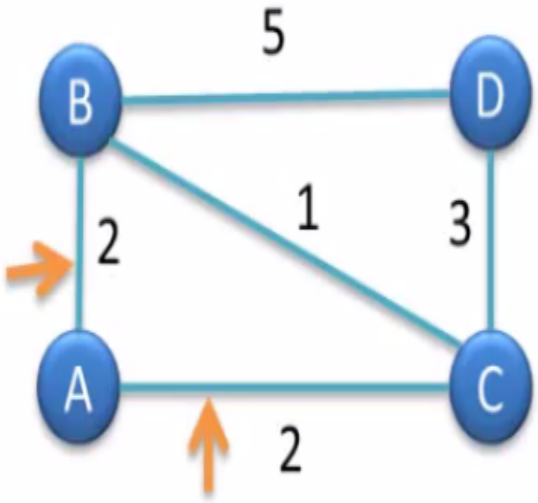
In this case edge BC is of least weight so we will select it.



Edge	BC				
Weight	1				

Now look at the graph again and find the edge of next minimum weight.

In this case edge AB and edge AC are having the minimum weight so we will select them both.

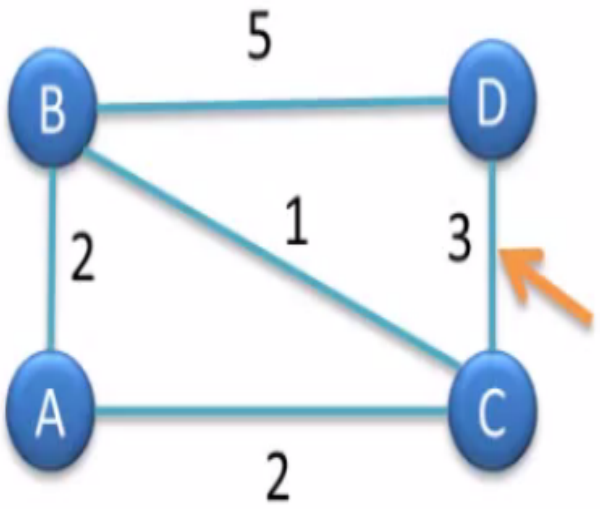


Note!
In case we have two or more edges with same weight then we can write them in any order in the edge table.

Edge	BC	AB	AC		
Weight	1	2	2		

Now look at the graph again and find the edge of next minimum weight.

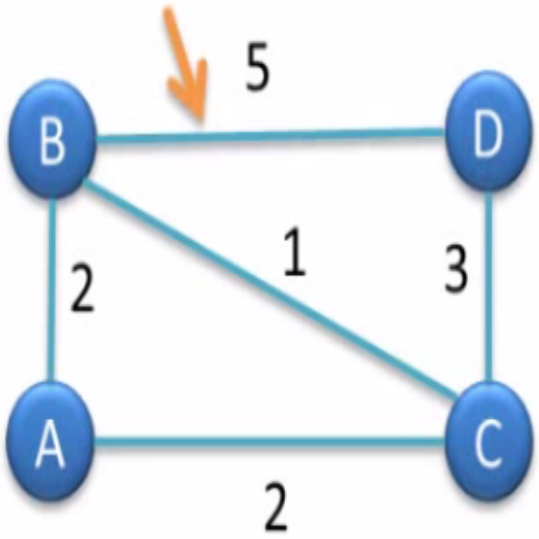
In this case edge DC has the minimum weight so we will select it.



Edge	BC	AB	AC	DC	
Weight	1	2	2	3	

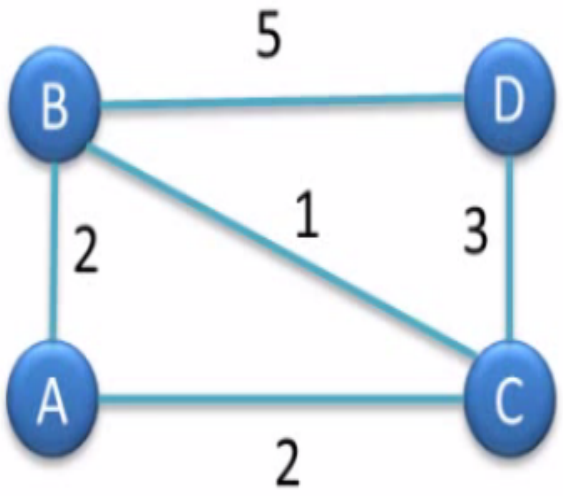
Now look at the graph again and find the edge of next minimum weight.

In this case edge BD has the minimum weight so we will select it.



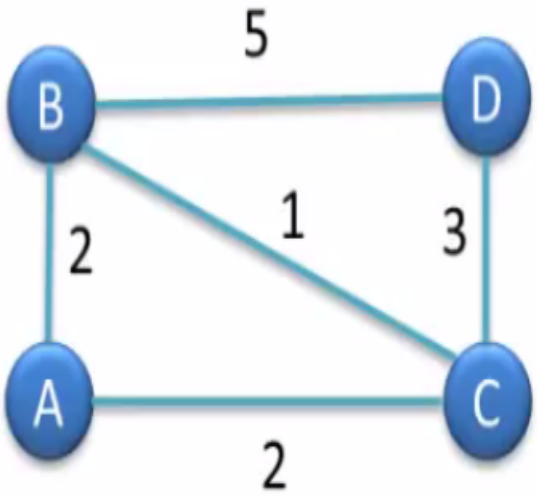
Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

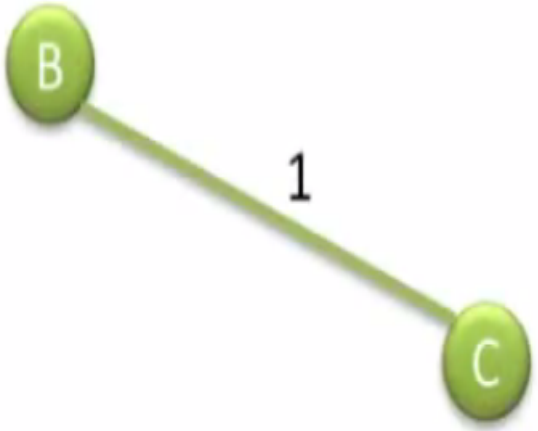
Time to find the minimum spanning tree.



Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

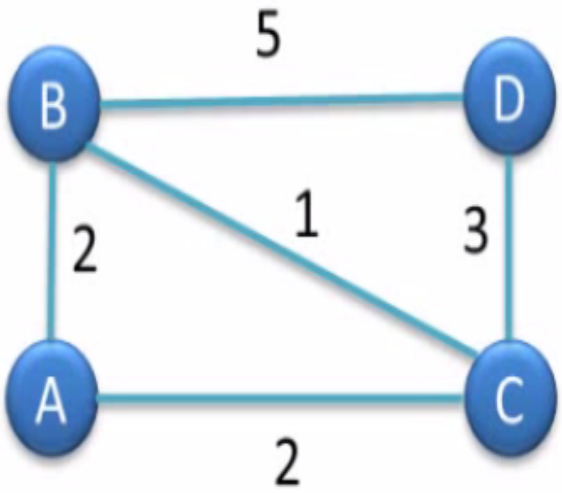
1 is the smallest weight so we will select edge BC

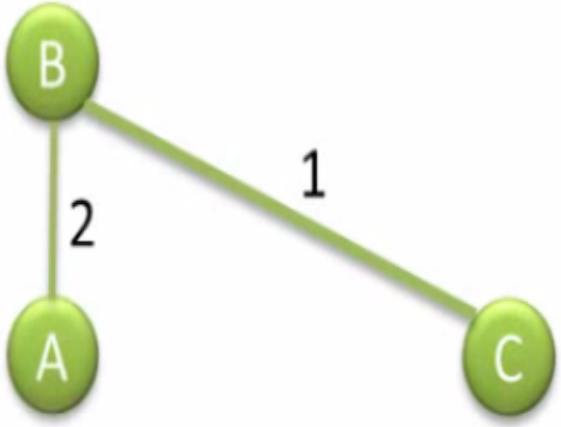




Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

2 is the next smallest weight and edge AB does not form a circuit with the previously selected edges so we will select it.



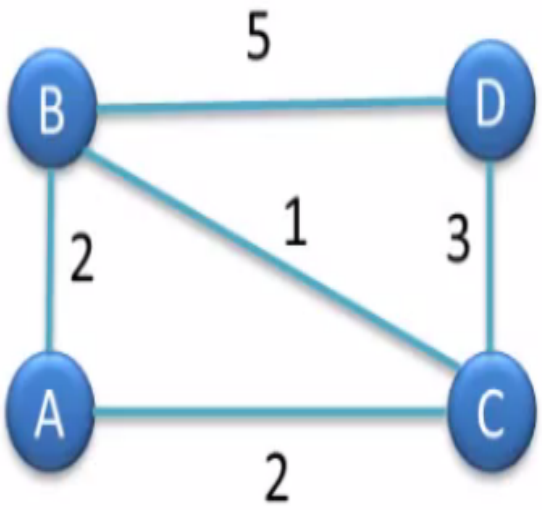


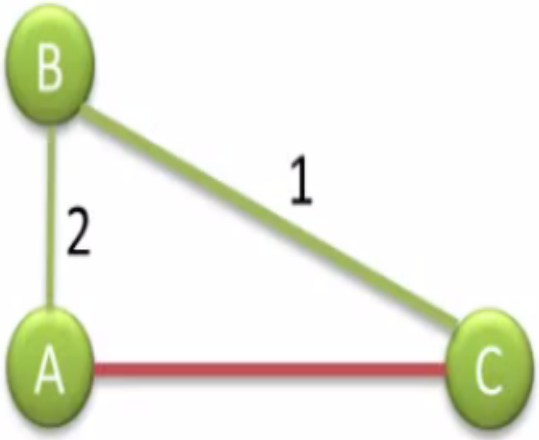
Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

2 is the next smallest weight.

But if we select edge AC then it will form a circuit.

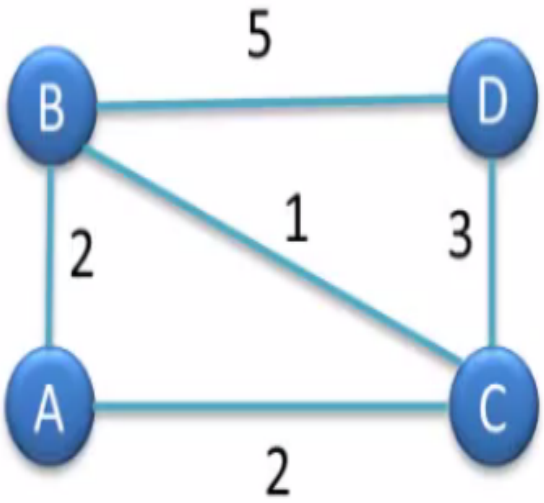
So we are going to reject edge AC.

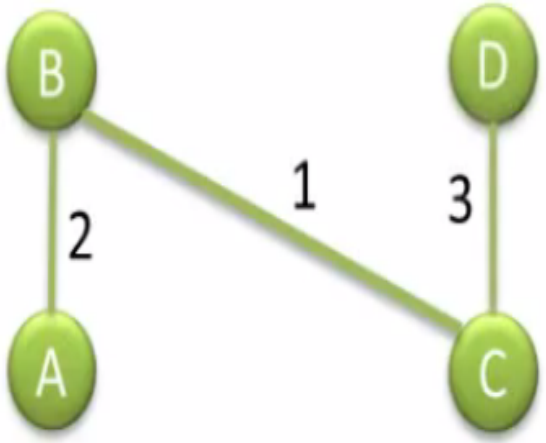




Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

3 is the next smallest weight and edge DC does not form a circuit with the previously selected edges. So we will select it.

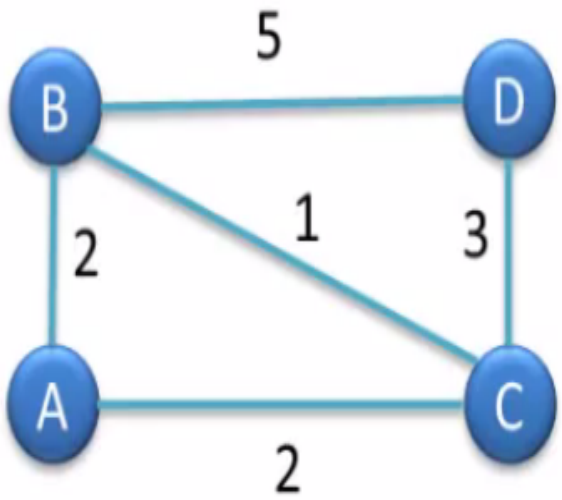


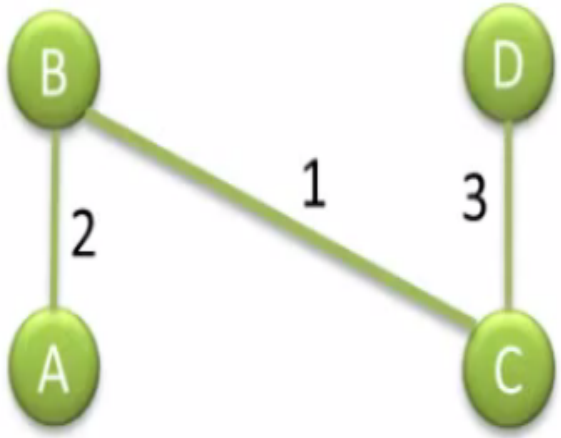


Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

Since we have got the 3 edges so we will stop here.

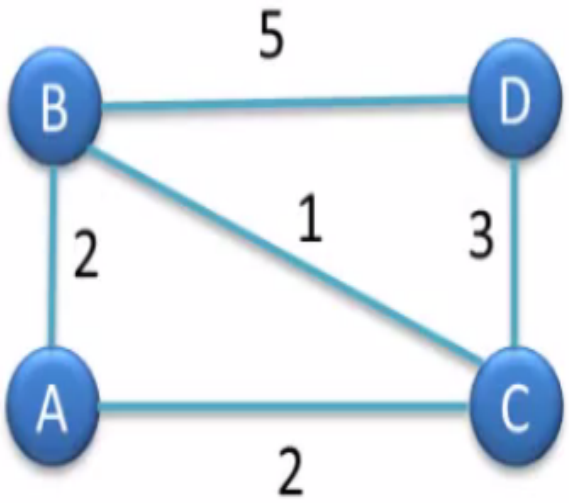
Note!
Our graph has 4 vertices so, our MST will have 3 edges.

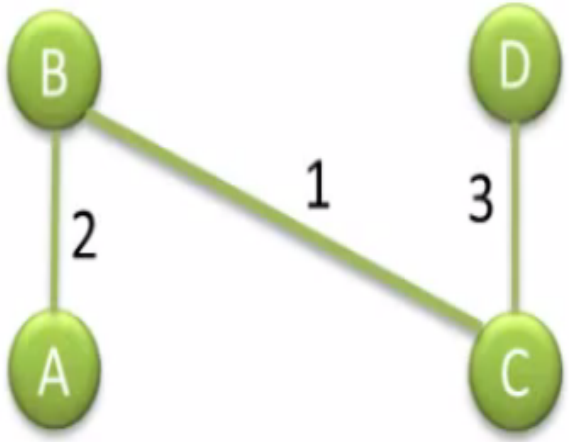




Edge	BC	AB	AC	DC	BD
Weight	1	2	2	3	5

Therefore, our required Minimum Spanning Tree is

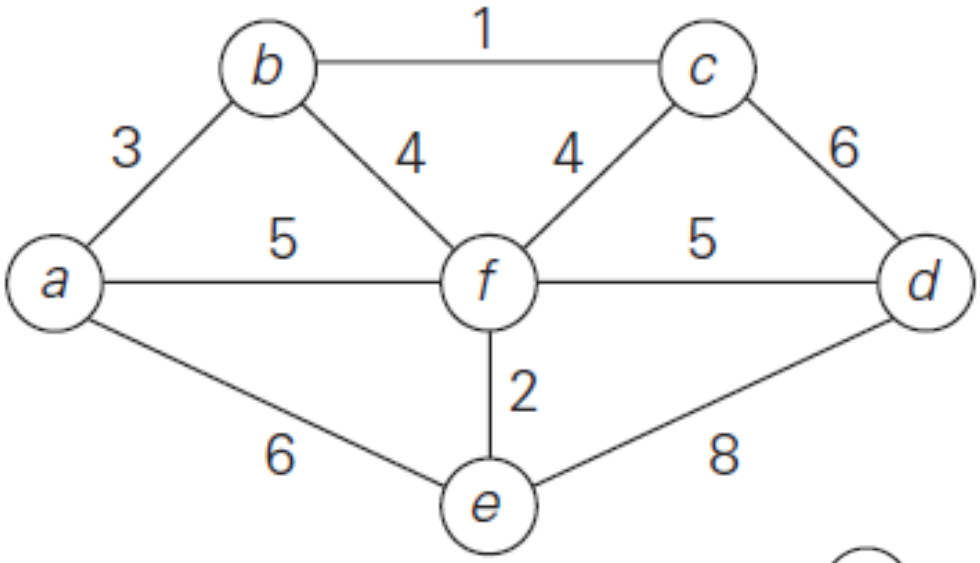


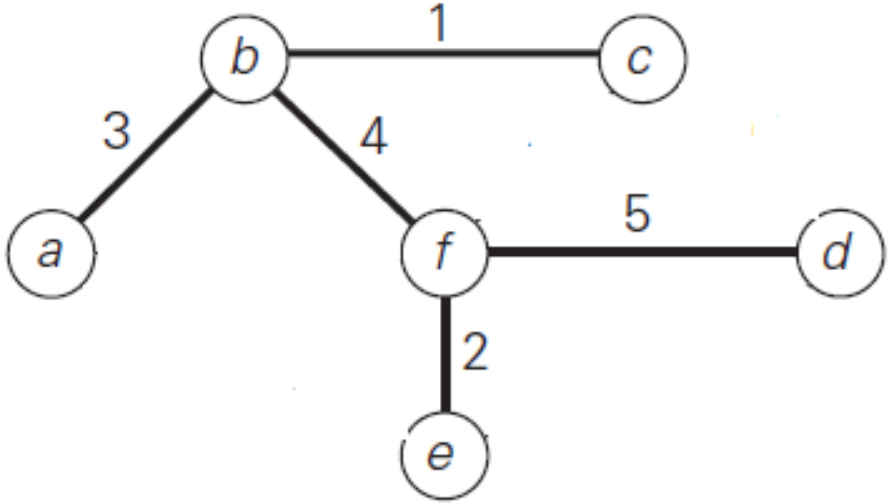


KRUSKAL'S ALGORITHM

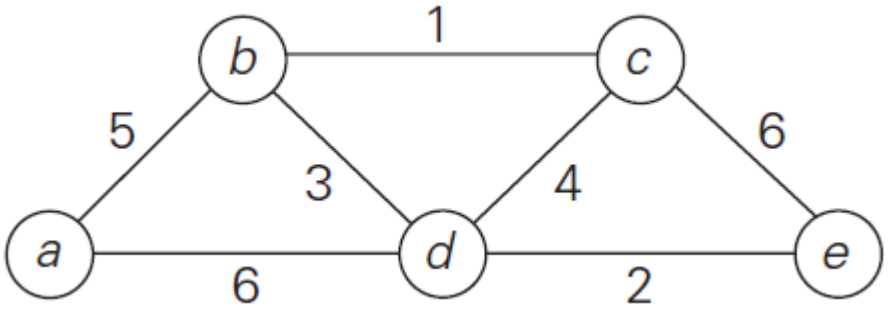
Kruskal's Algorithm (G, c)
Sort the edges in order of increasing cost.
Initially $T = \emptyset$
For each edge $e = (v, w)$ in the sorted order
    If there is currently no path from v to w in (V, T) then
        (Adding e won't create a cycle)
        Add e to T.
    EndIf
EndFor
Return the set of edges T

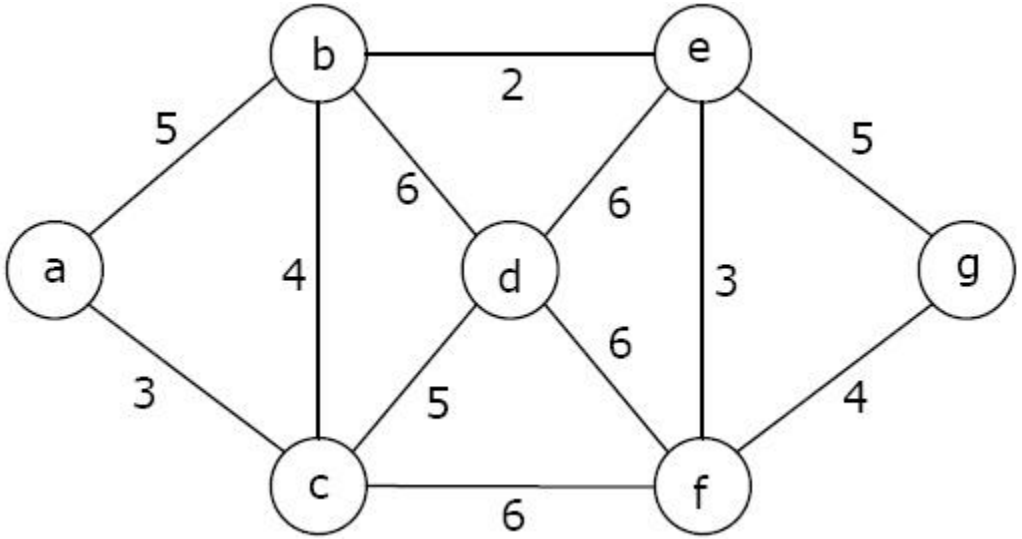
KRUSKAL'S ALGORITHM





APPLY PRIM'S & KRUSKAL'S





THANK YOU