Exercises 2 -ch1

Ex. 1.13: Find the weight of the minimal spanning tree for the following network.

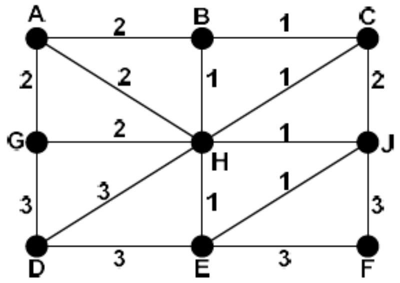


Figure 1.47: A connected graph.

Solutions:

We have weighted connected undirected graph with $G = (V, E)$, $|V|$ = n= 9, so the spanning tree must have only n-1= **8** edges.

By use Nearest Neighbour Algorithm (NNA).

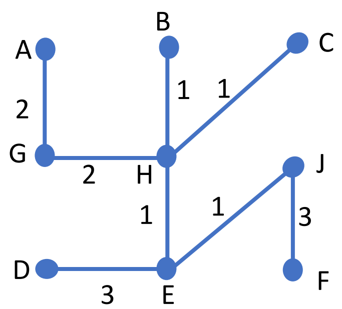
Edges	weights	delete	Edges	weights	delete
DH	3	yes	**HG**	**2**	no
DG	3	yes	**GA**	**2**	no
DE	**3**	no	**BC**	1	yes
EF	3	yes	**BH**	**1**	no
FJ	**3**	no	**HC**	**1**	no
JC	2	yes	**HJ**	1	yes
BA	2	yes	**HE**	**1**	no
AH	2	yes	**EJ**	**1**	no

Step 1: Arrange the edges of G in the order of decreasing weights.

Step 2: Proceeding sequentially, deletes each edge that does not disconnect the graph until $n - 1$ edges remain.

Step 3: Exit.

According to NNA the weight of the minimal spanning tree is **14** and is given by,



By use Brute-Force method (BFM).

It is difficult to solve manually….

Ex. 1.16: Find the weight of the minimal spanning tree for the following network.

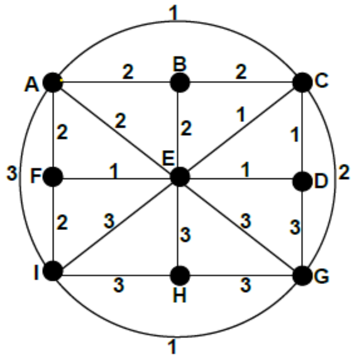


Figure 1.53: A connected graph.

Solutions:

We have weighted connected undirected graph with $G = (V, E)$, $|V| = 9$, so the spanning tree must have only n-1= **8** edges.

By use Kruskal's Algorithm.

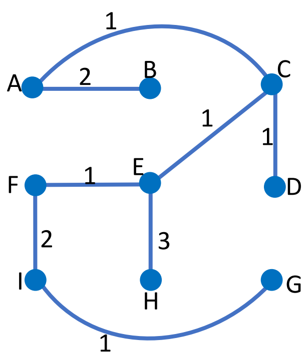
i	Edges	weights	Add	i	Edges	weights	Add
1	**AC**	1	yes	11	**BC**	2	no
2	**CE**	1	yes	12	**BE**	2	no
3	**CD**	1	yes	13	**CG**	2	no
4	**DE**	1	no	14	**AI**	3	no
5	**EF**	1	yes	15	**IE**	3	no
6	**GI**	1	yes	16	**EH**	3	yes
7	**AB**	2	yes	17	**HI**	3	no
8	**AE**	2	no	18	**HG**	3	no
9	**AF**	2	no	19	**GE**	3	no
10	**FI**	2	yes	20	**GD**	3	no

Step 1: Arrange the edges of G in order of increasing weights.

Step 2: Starting only with the vertices of G and proceeding sequentially, add each edge which does not result in a cycle until $n - 1$ edges are added.

Step 3: Exit.

The minimal spanning tree has weight **12** and is given by,

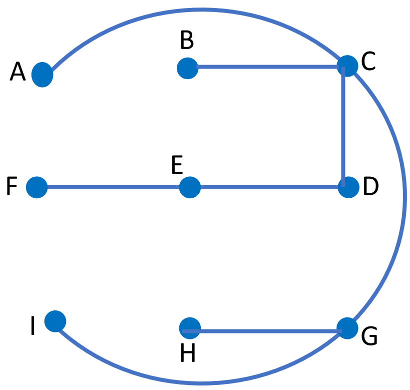


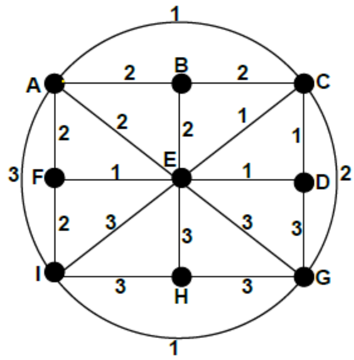
By use Prim's Algorithm.

Let us select the start vertex $\boldsymbol{A}$.

Step 1: Select an arbitrary vertex from the graph G and add it to a tree T.
Step 2: Consider the weights of each edge connecting to the vertices in T and select the minimum.
Step 3: Repeat step 2 until $n-1$ edges are added to the tree T.
Step 4: Exit.

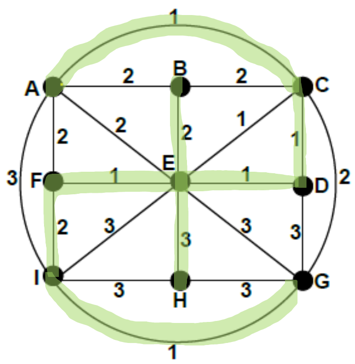
Iteration	Tree	Minimum edge	Minimum weight
0	{A}	AC	1
1	{A,C}	CD or CE	1
2	{ A,C,D}	DE or CE	1
3	{A,C,D,E}	EF	1
4	{A,C,D,E,F}	CG,FI,CB,EB, or AB	2
5	{A,C,D,E,F,G}	GI	1
6	{A,C,D,E,F,G,I}	CB,AB, or EB	2
7	{A,C,D,E,F,G,I,B}	GH,IH, or EH	3
Total	{A,C,D,E,F,G,I,B,H}	---	**12**





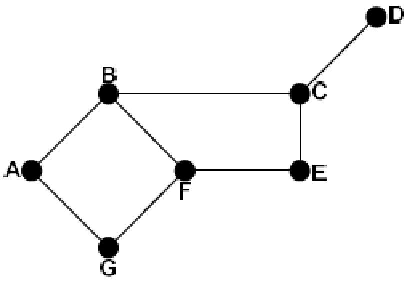
By use Boruvka's Algorithm.

Component	Closest weight edge	Weight
{A}	AC	1
{B}	BE (or BC or BA)	2
{C}	CD (or CE or CA)	1
{D}	DE (or DC)	1
{E}	EF (or ED)	1
{F}	FI	2
{G}	GI	1
{H}	HE (or HG or HI)	3
{I}	--	--



The minimal spanning tree has weight **12 .**

Ex. 1.18: Find **BFS** spanning tree of the following graph. Start at vertex A.



Solutions:

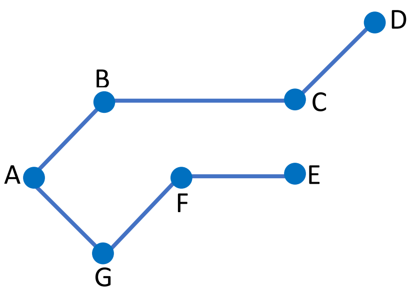
We have connected undirected graph with $G = (V, E)$, $|V| = 7$, so the spanning tree must have only n-1= **6** edges.

Start at vertex A.

Starting vertex	Adjacent vertices (not visited yet)	Visited vertex	FIFO-queue
A	B,G	A	B,G
B	F,C	B	G,F,C
G	F	G	F,C
F	E	F	C,E
C	D,E	C	E,D
E	-	E	D
D	-	D	-

BFS algorithmic steps:
Step 1: Start at some vertex. Mark it as a visited vertex.
Step 2: Search on all adjacent vertices to the visited vertex. Add the non-visited adjacent vertices in the FIFO queue.
Step 3: Pull out the first non-visited vertex from the FIFO-queue and traverse to it.
Step 4: Go back to step 1 till all vertices are visited.

So the order by which the vertices are visited is A,B,G,F,C,E and D. Then the spanning tree becomes,



Ex. 1.21: Find **DFS** tree of the following graph. Start at vertex A.

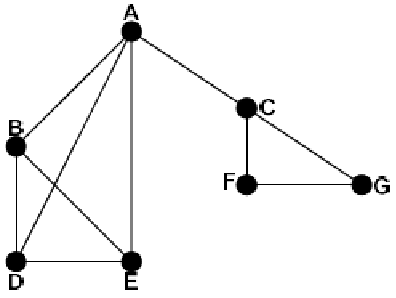


Figure 1.63: A connected graph.

Solutions:

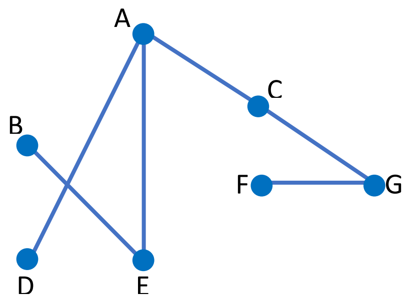
We have connected undirected graph with $G = (V, E)$, $|V| = 7$, so the spanning tree must have only n-1= **6** edges.

Start at vertex A.

Starting vertex	Adjacent vertices (not visited yet)	Visited vertex	LIFO-stack
A	B,E,D,C	A	B,E,D,C
C	F,G	C	B,E,D,F,G
G	F	G	B,E,D,F
F	-	F	B,E,D
D	E,B	D	B,E
E	B	E	B
B	-	B	-

DFS algorithmic steps:
Step 1: Start at some vertex. Mark it as a visited vertex.
Step 2: Search on all adjacent vertices to the visited vertex. Add the non-visited adjacent vertices in the LIFO stack.
Step 3: Select the top vertex in the LIFO-stack and traverse to it.
Step 4: Go back to step 1 till all vertices are visited.

So the order by which the vertices are visited is A,C,G,F,D,E and B. Then the spanning tree becomes,



H.W

Ex. 1.14: Find the weight of the minimal spanning tree for the following network.

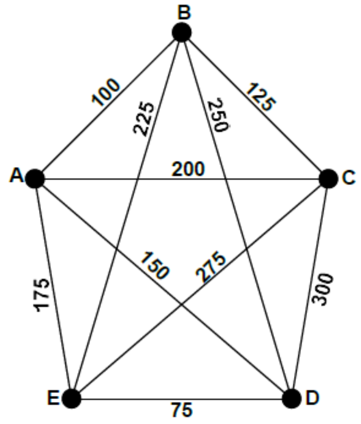


Figure 1.49: A connected graph.

Ex. 1.15: Find the weight of the minimal spanning tree for the following network.

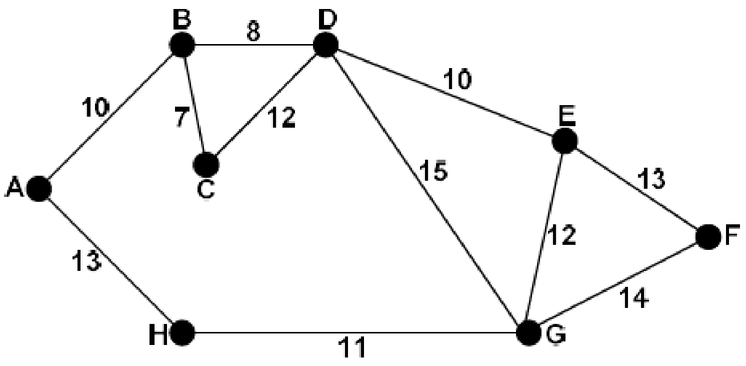


Figure 1.51: A connected graph.

Ex. 1.19: Find **BFS** spanning tree of the following graph. Start at vertex A.

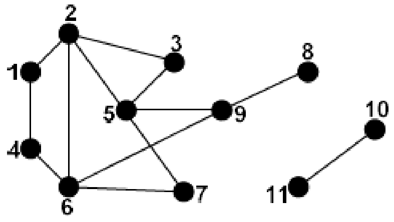


Figure 1.60: A disconnected graph.