



EMTH403

Mathematical Foundation for Computer Science

Nitin K. Mishra (Ph.D.)

Associate Professor

Lecture Outcomes

After this lecture, you will be able to

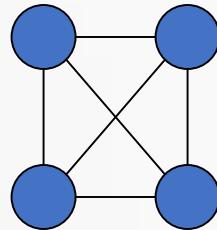
- understand what Spanning Trees.
- understand what is Prim's Algorithm.

Spanning Trees

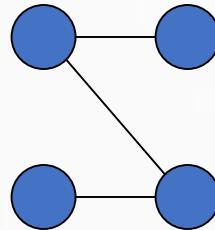
A spanning tree of a graph is just a subgraph that contains all the vertices and is a tree.

A graph may have many spanning trees.

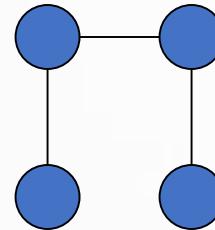
Graph A



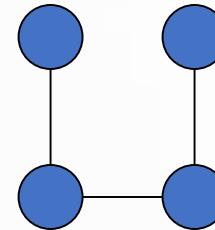
Some Spanning Trees from Graph A



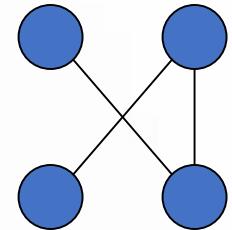
o
r



o
r

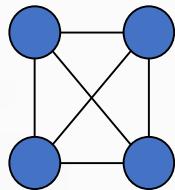


o
r

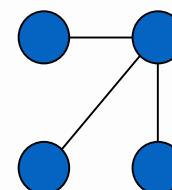
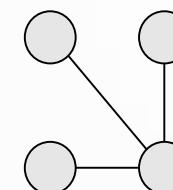
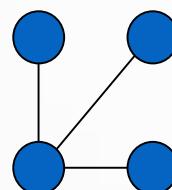
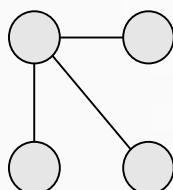
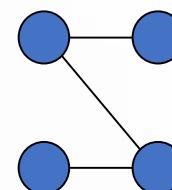
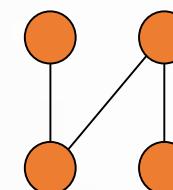
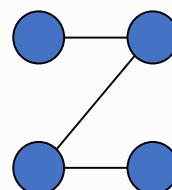
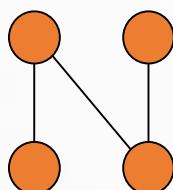
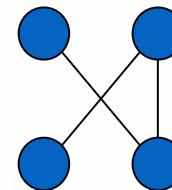
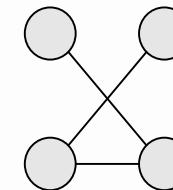
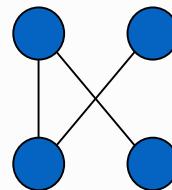
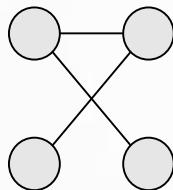
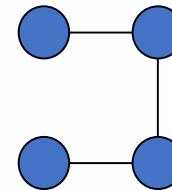
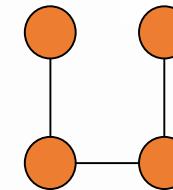
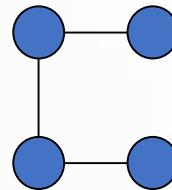
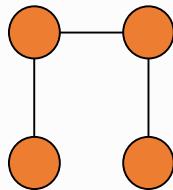


Spanning Trees

Complete Graph



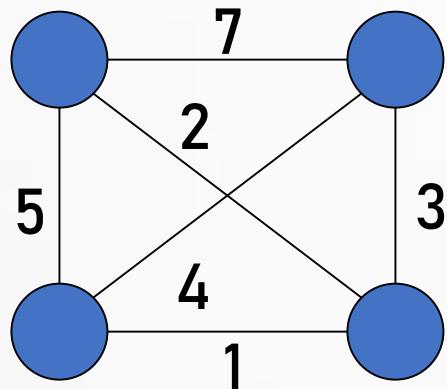
All 16 of its Spanning Trees



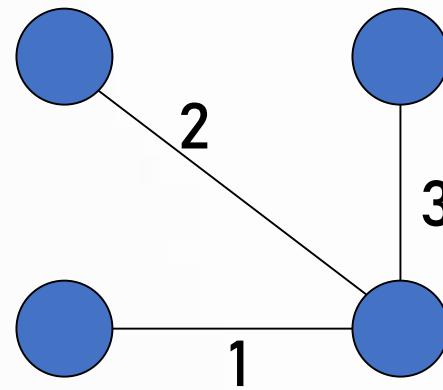
Minimum Spanning Trees

The Minimum Spanning Tree for a given graph is the Spanning Tree of minimum cost for that graph.

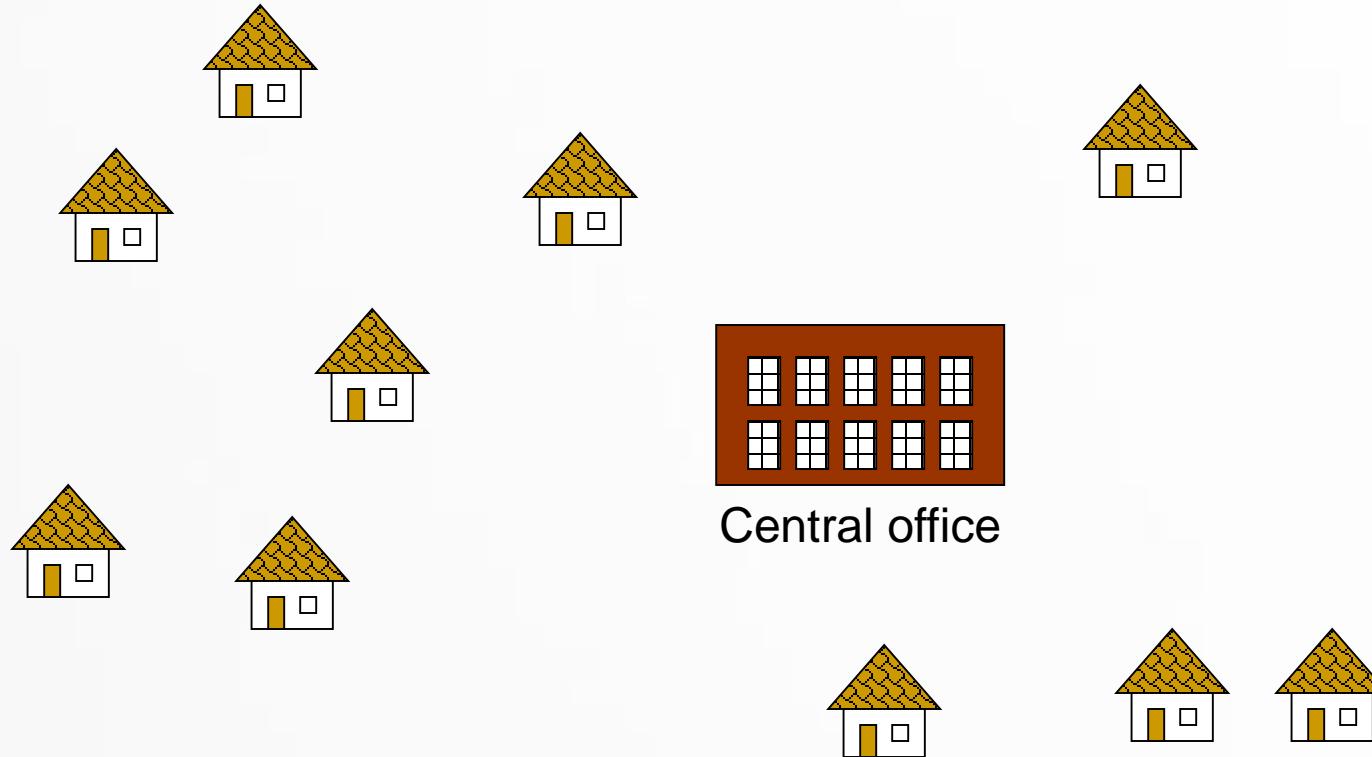
Complete Graph



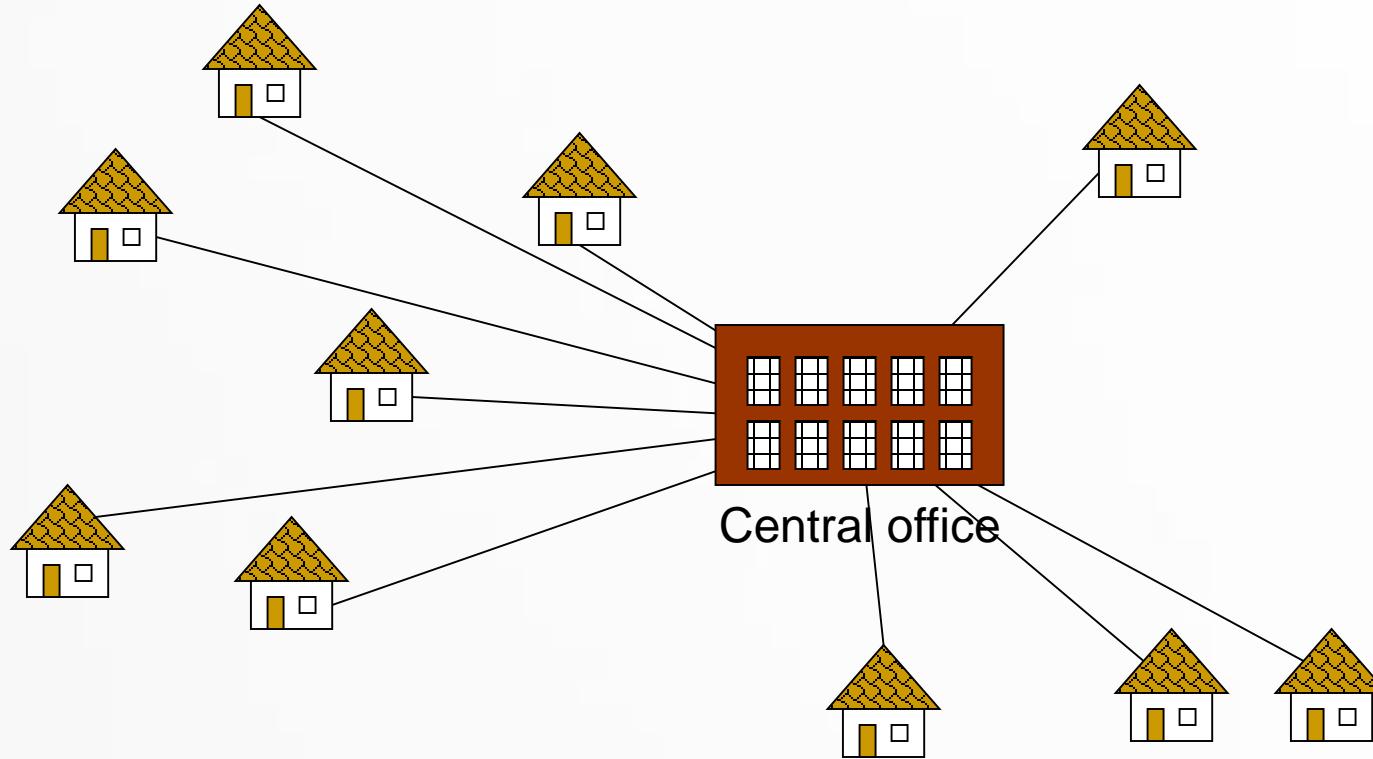
Minimum Spanning Tree



Spanning Trees

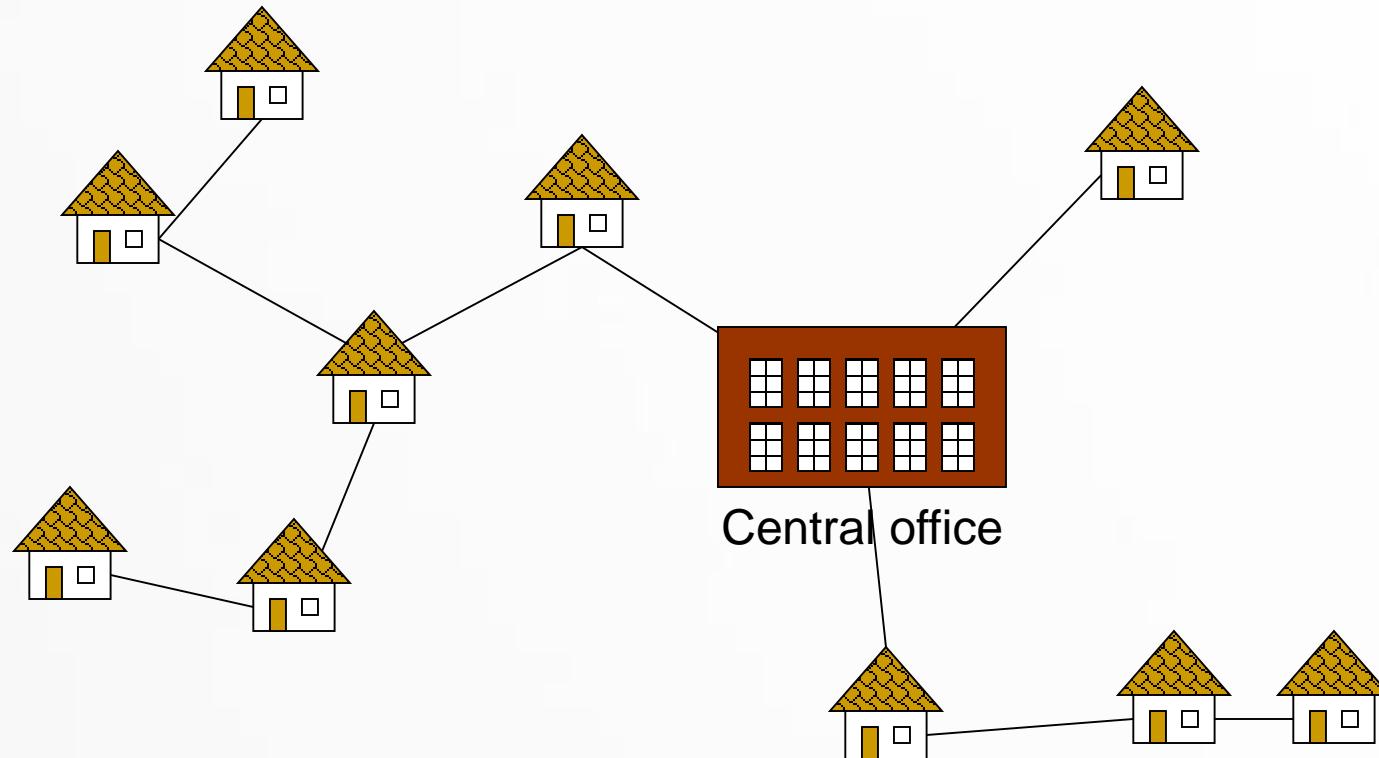


Spanning Trees



Expensive!

Spanning Trees



Minimize the total length of wire connecting the customers.

Algorithms for Obtaining the Minimum Spanning Tree

Kruskal's Algorithm

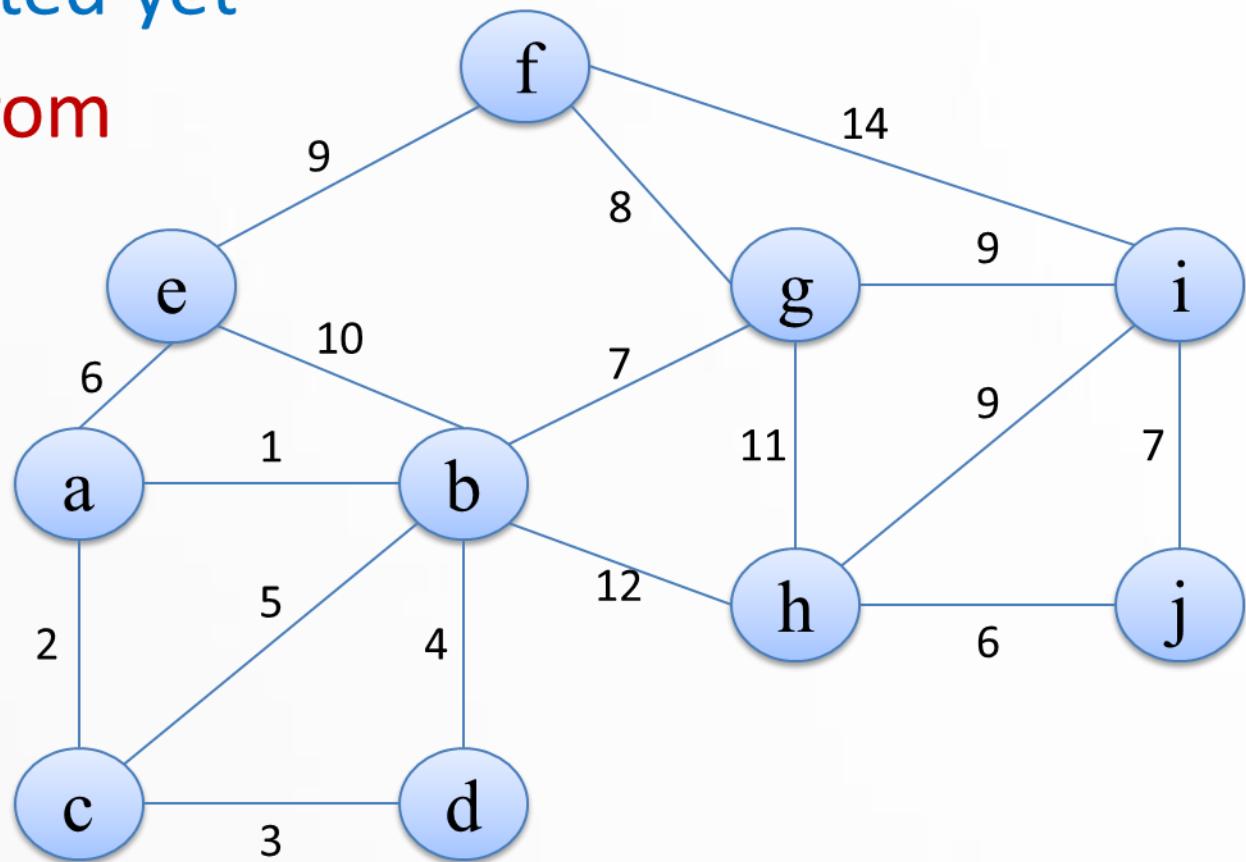
Prim's Algorithm

Spanning Trees

- Prim's algorithm: Start with any *one node* in the spanning tree, and repeatedly add the cheapest edge, and the node it leads to, for which the node is not already in the spanning tree.
 - Here, we consider the spanning tree to consist of both nodes and edges

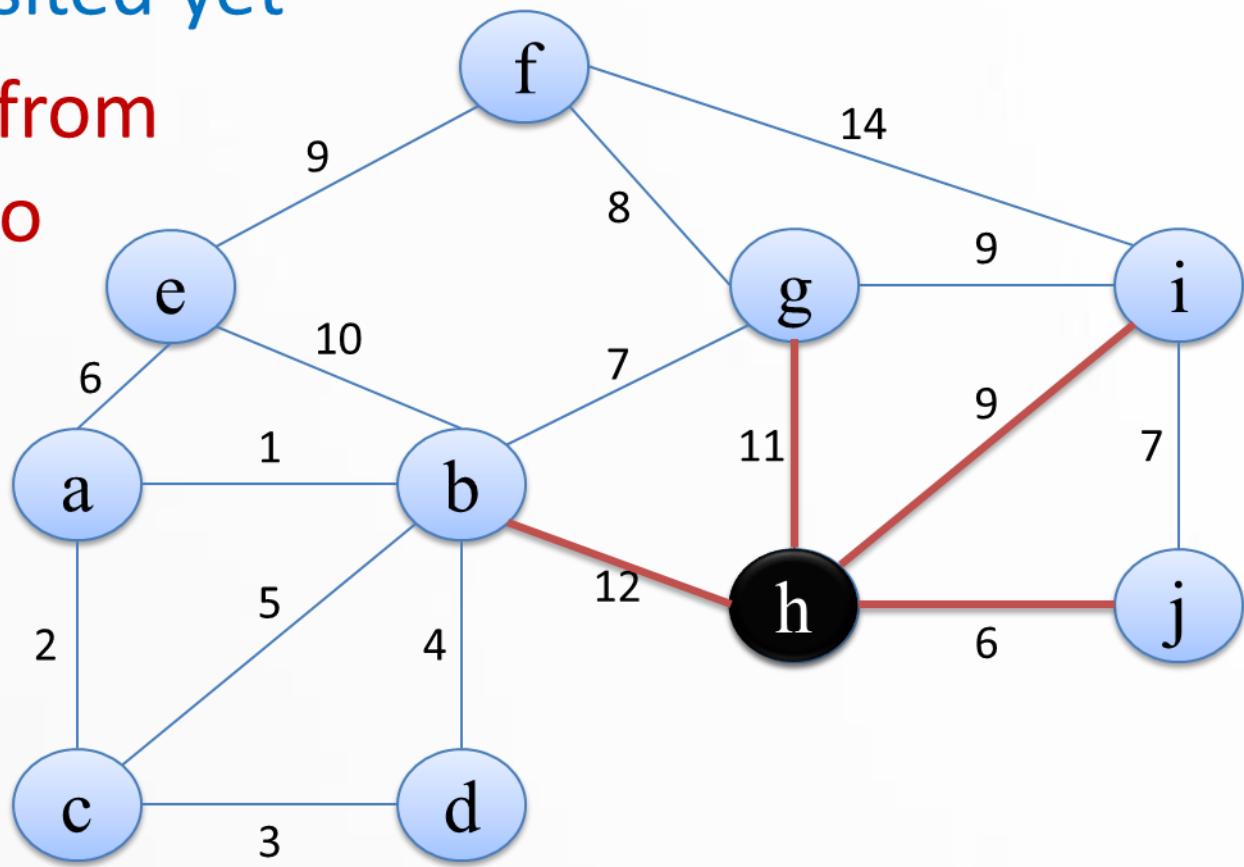
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



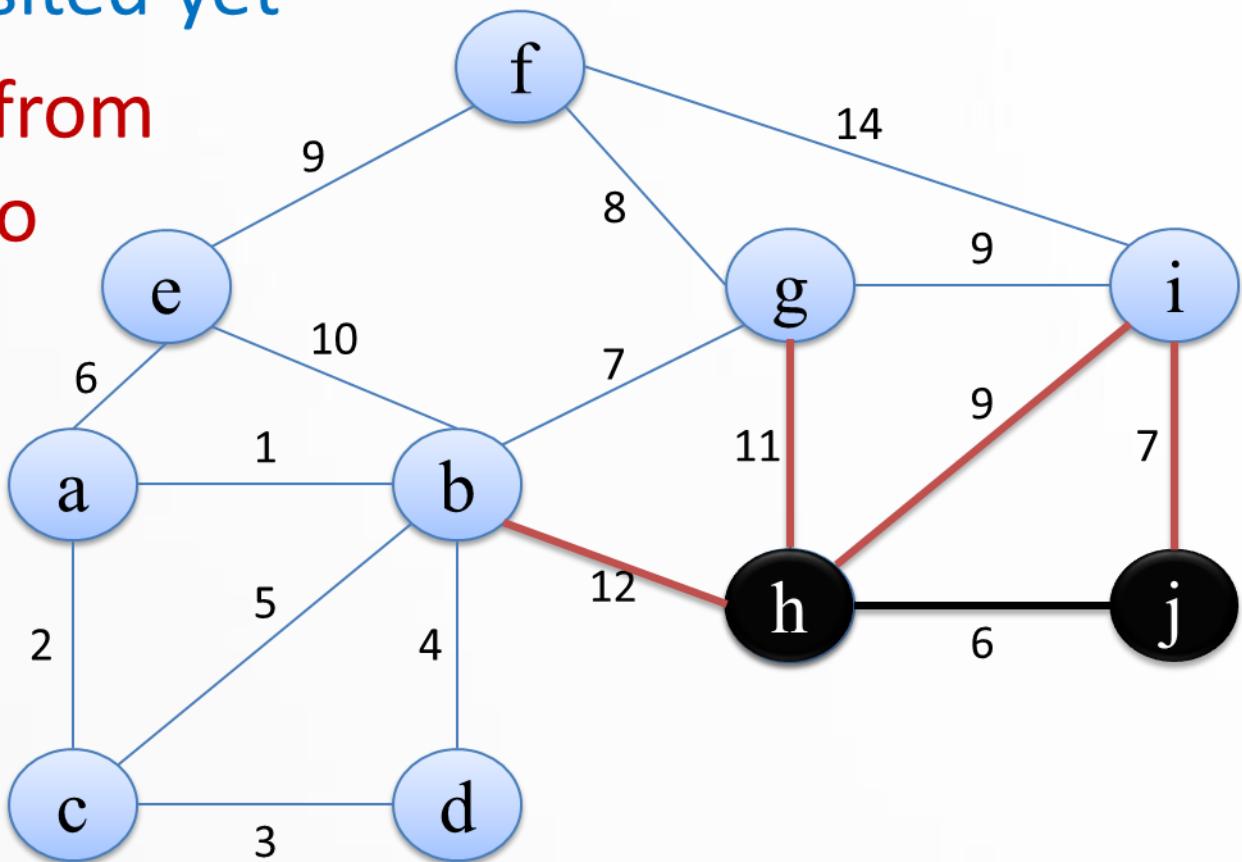
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



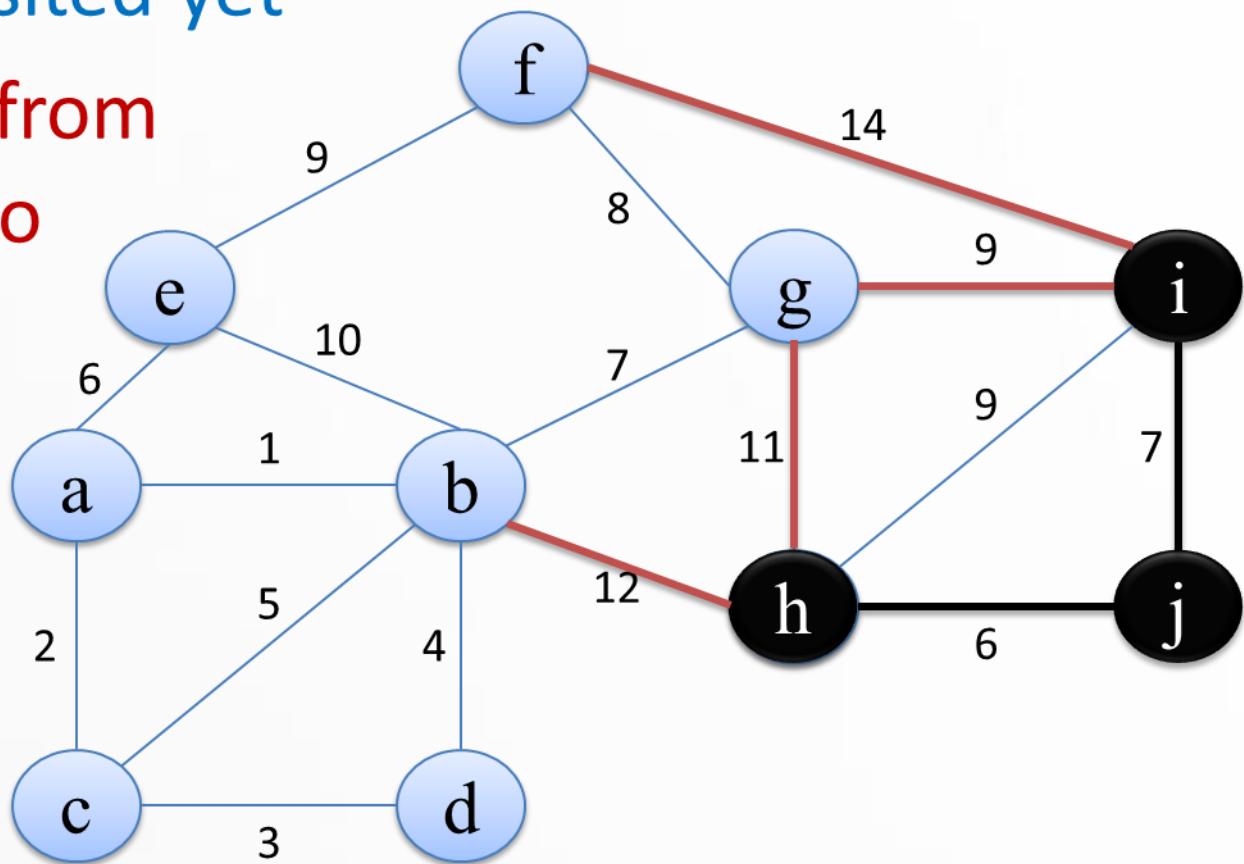
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



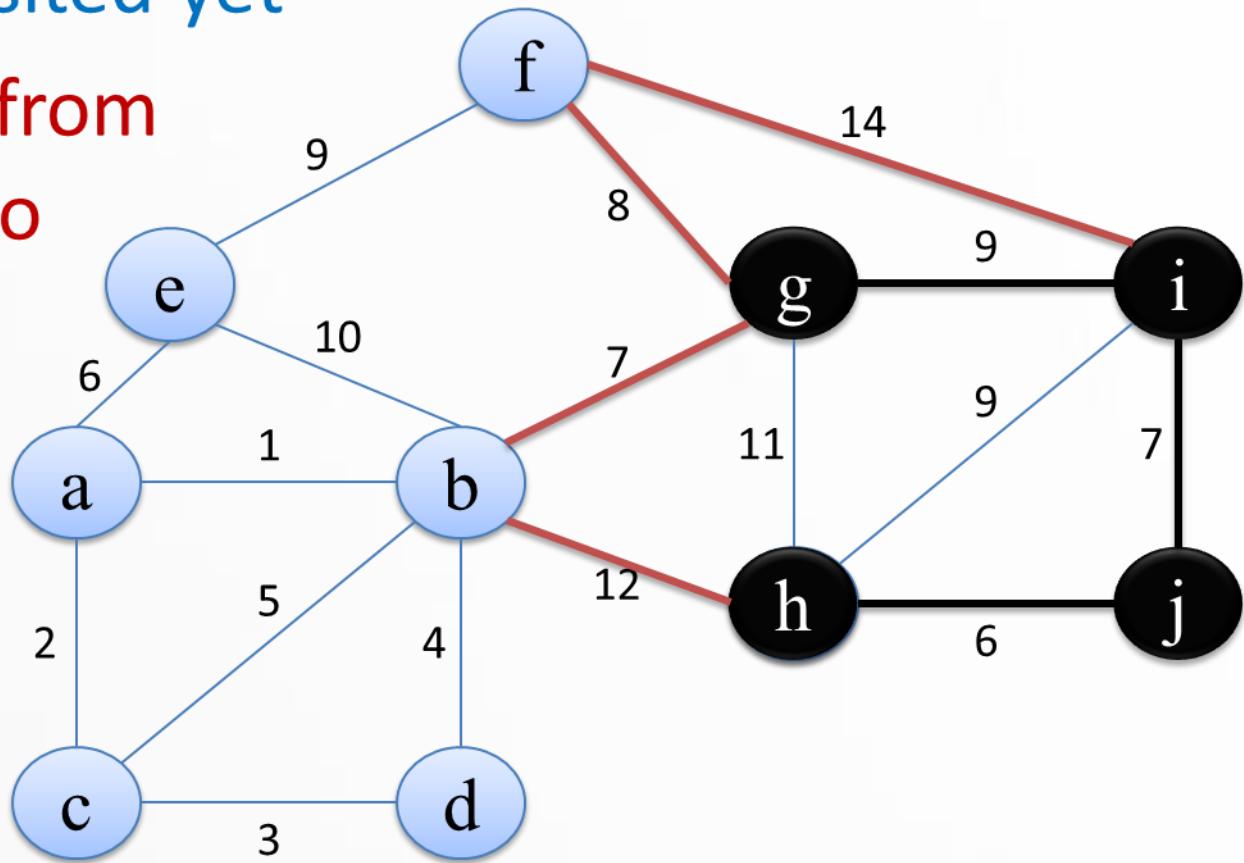
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



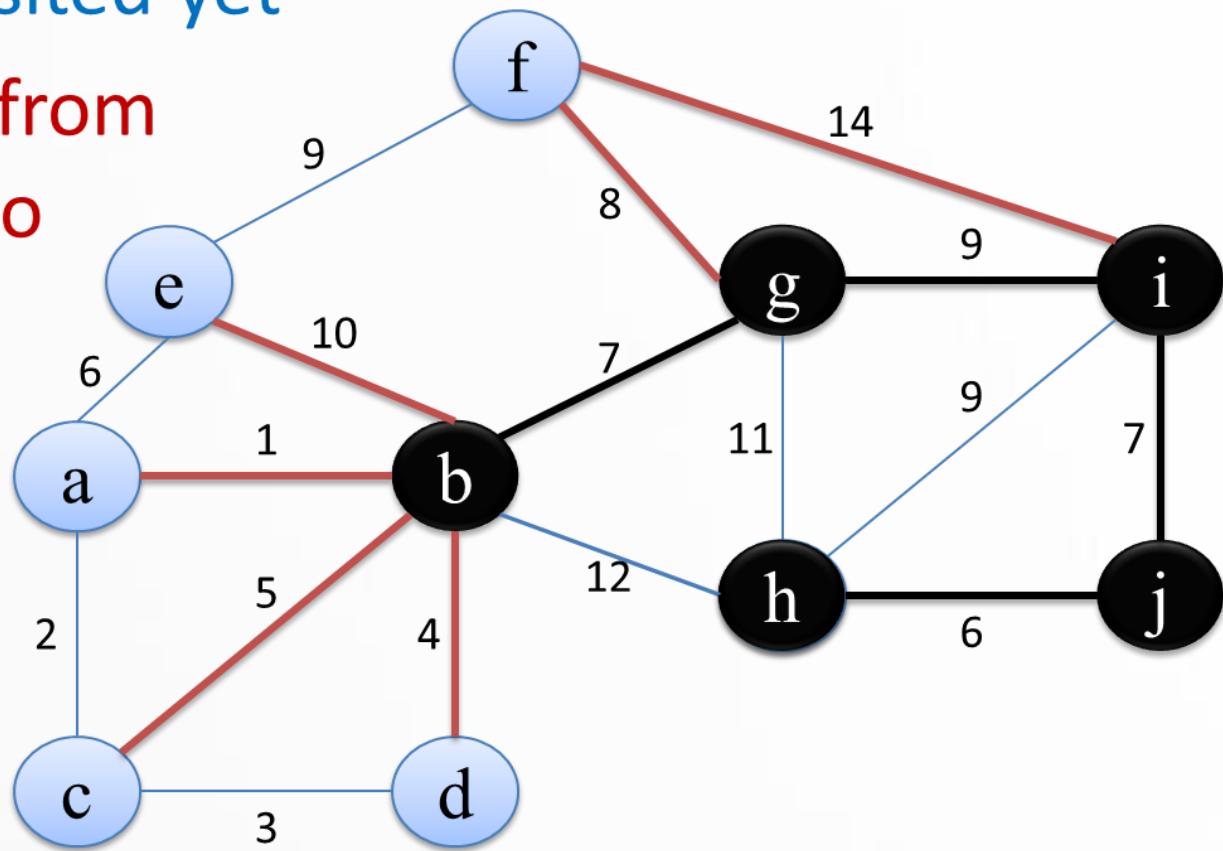
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



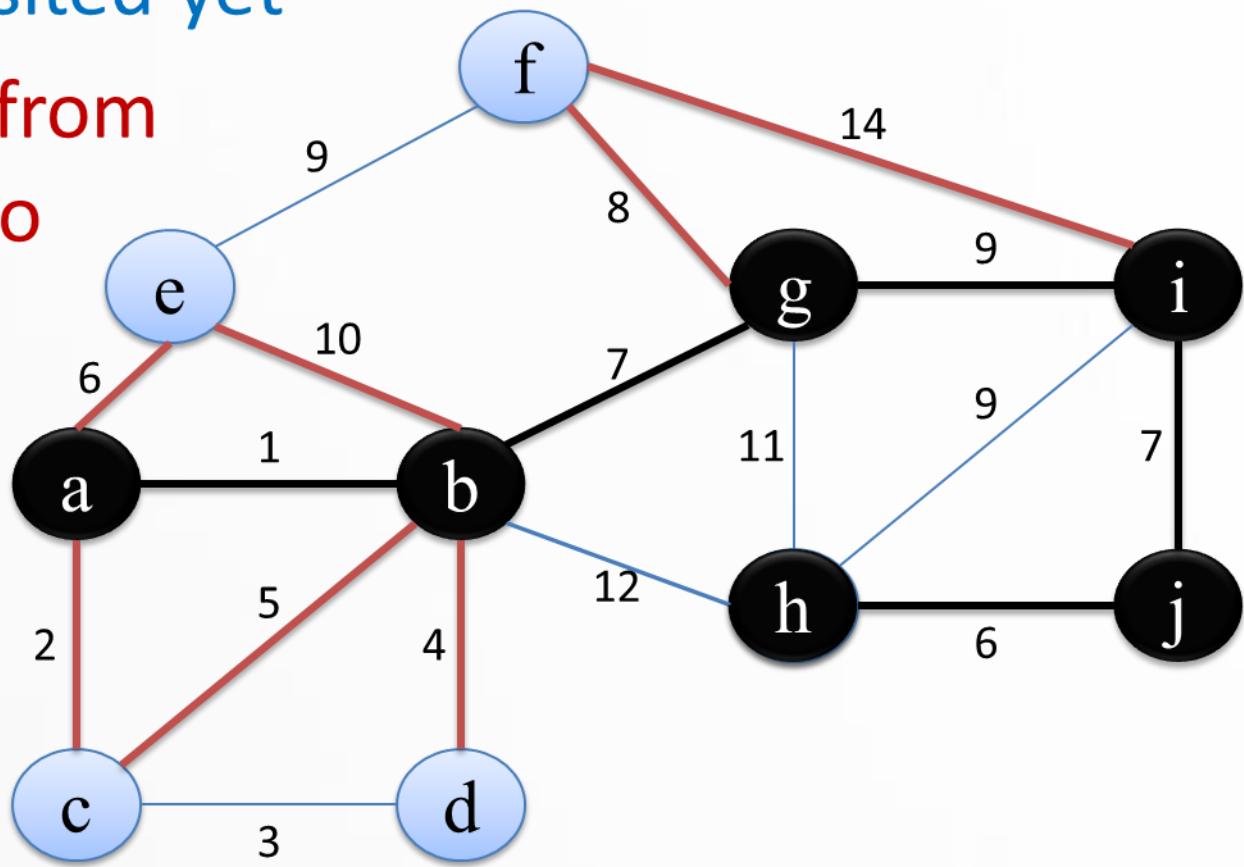
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



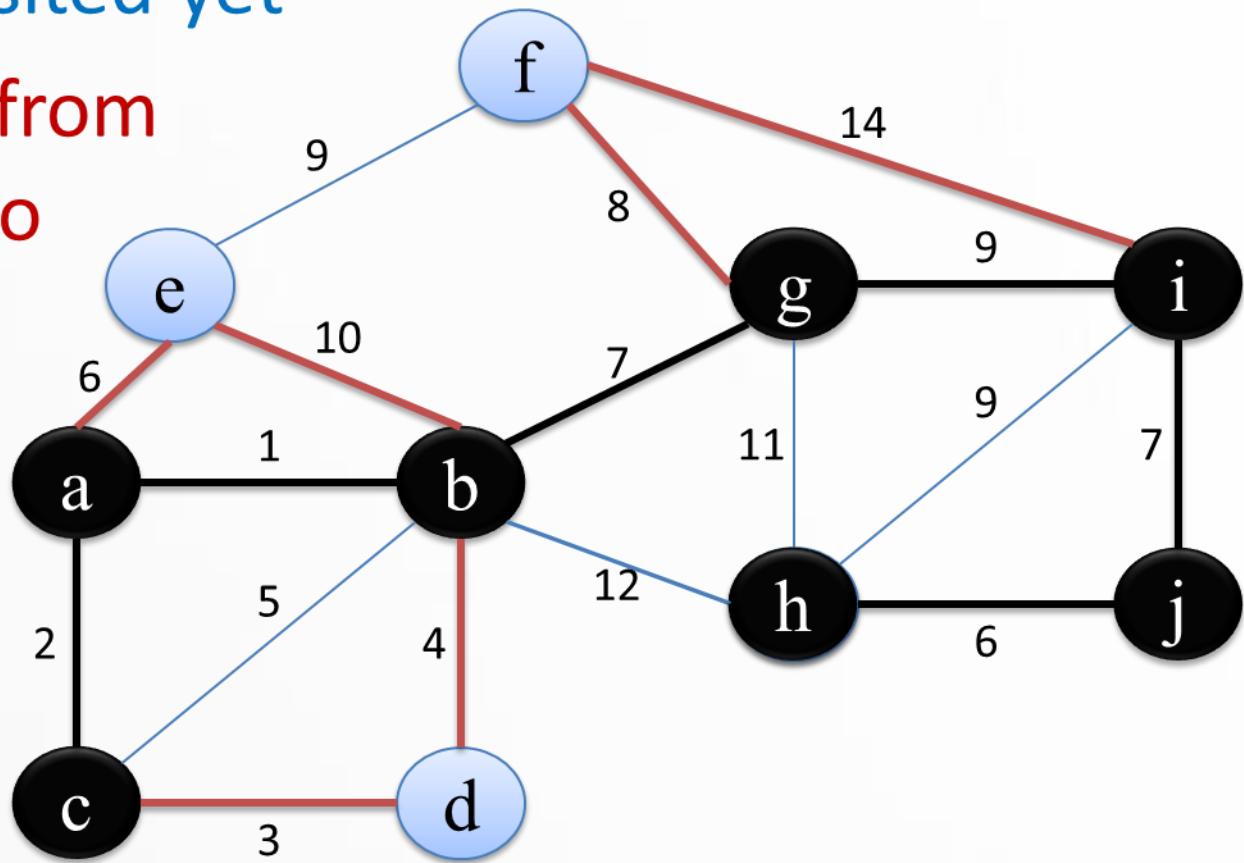
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



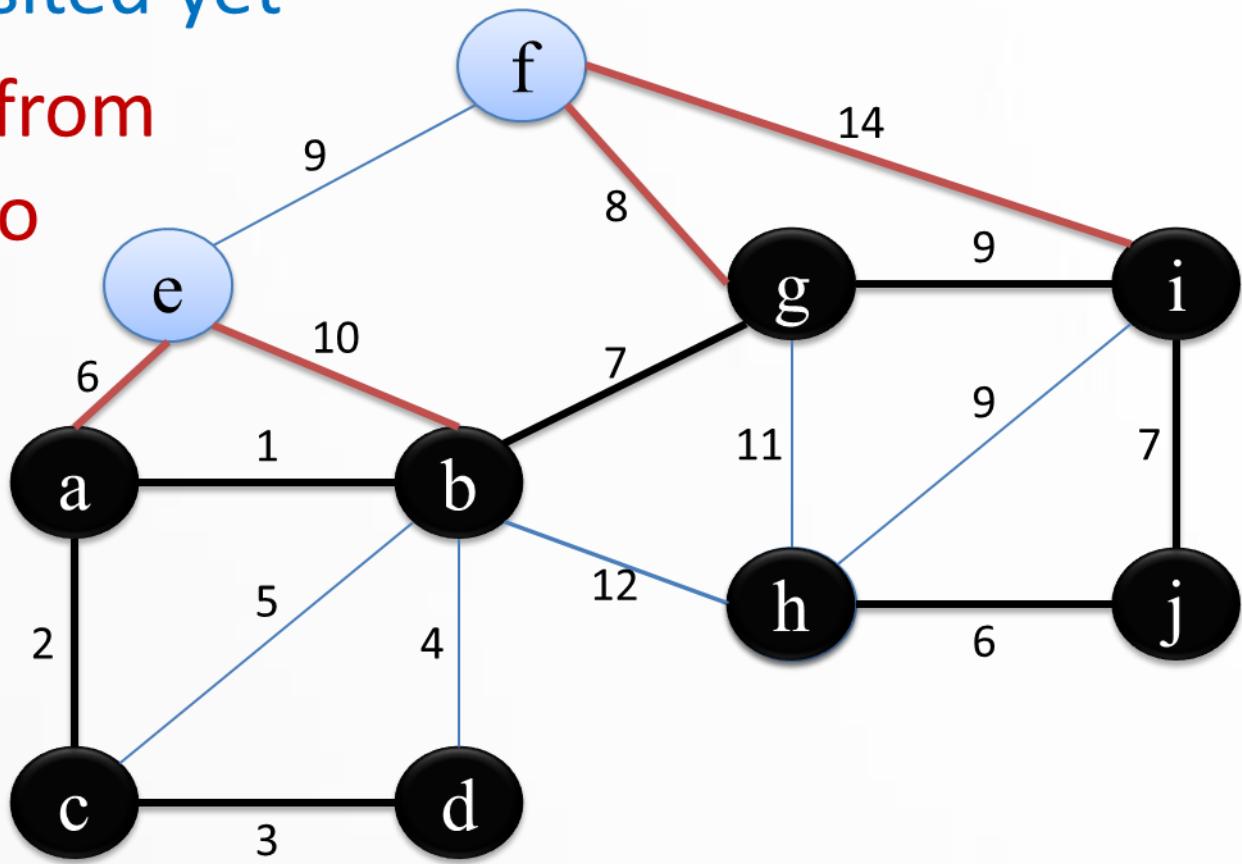
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



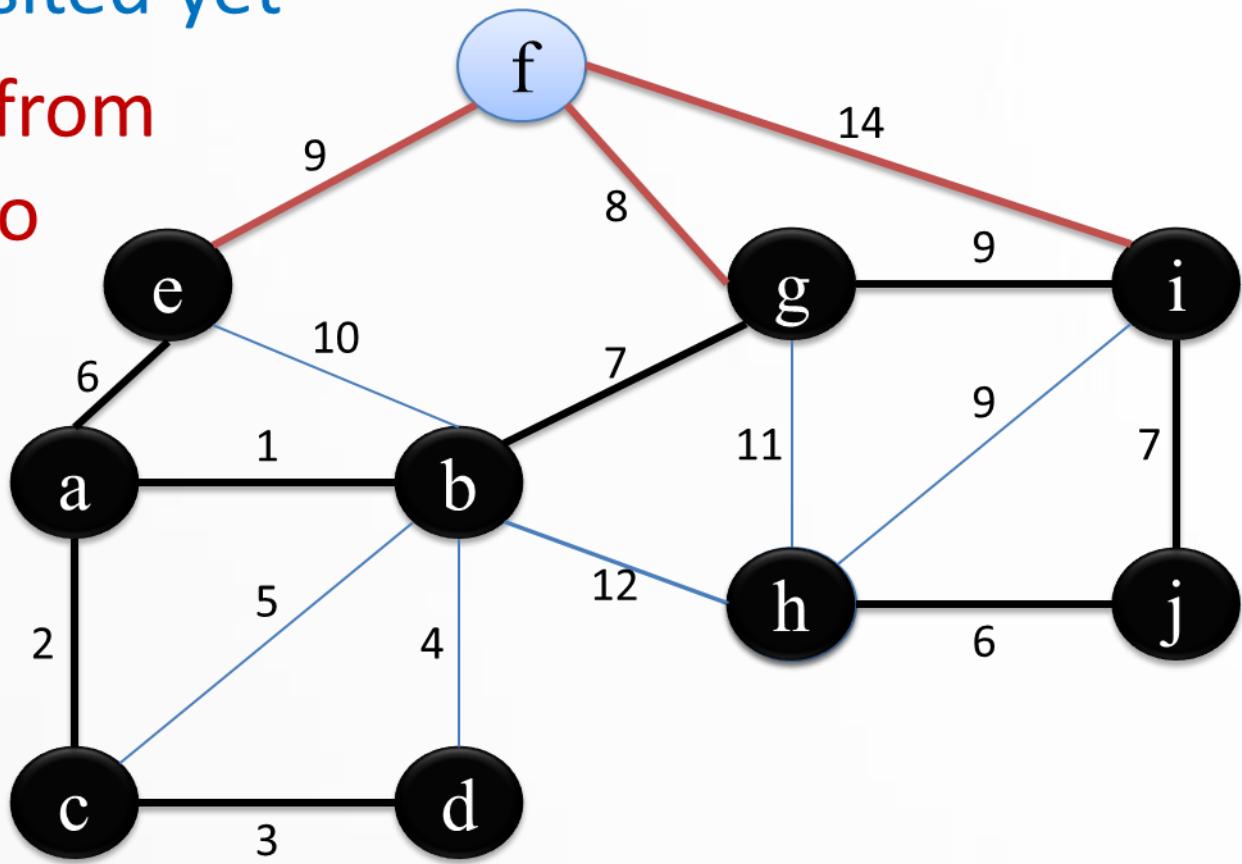
Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



Prim's algorithm

- Start at an arbitrary node, say, h.
- **Blue:** not visited yet
- **Red:** edges from nodes $\in T$ to nodes $\notin T$
- **Black:** in T



Prim's algorithm

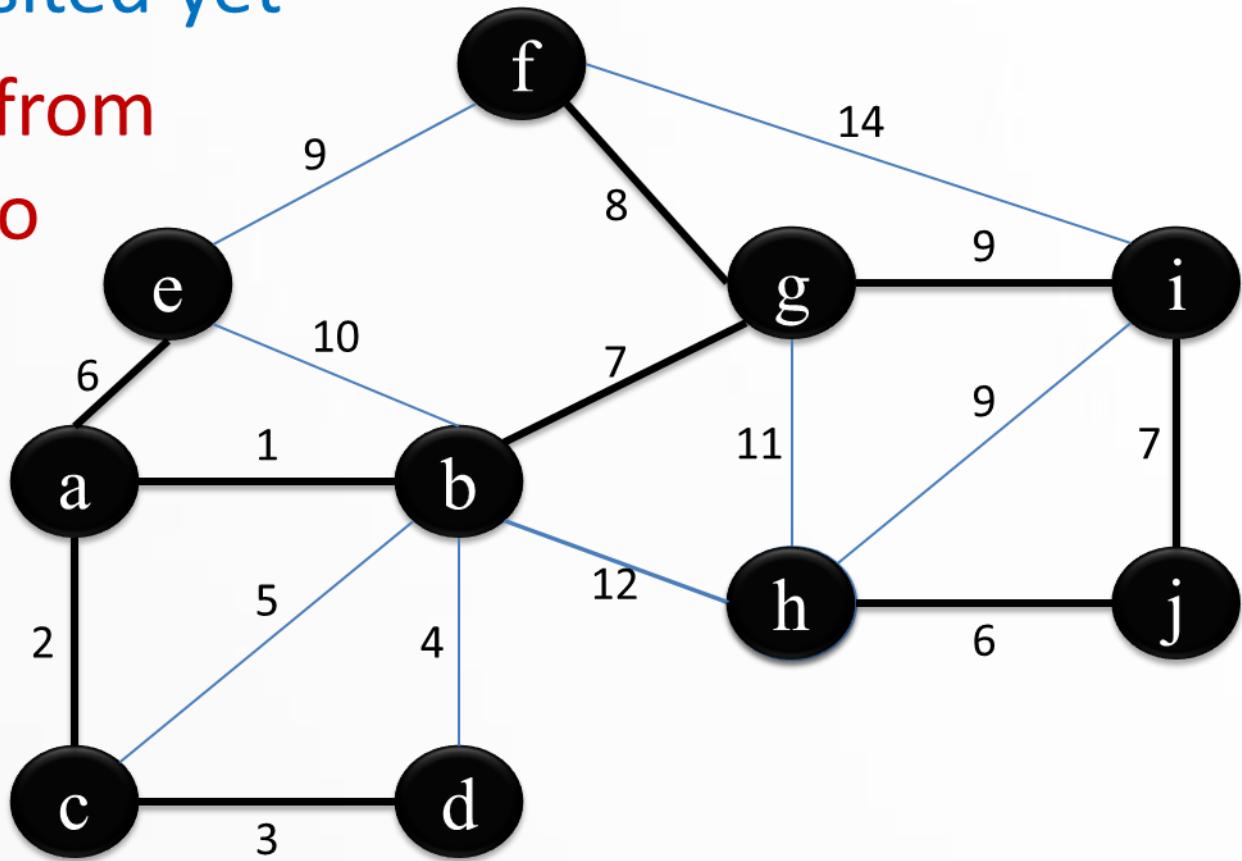
- Start at an arbitrary node, say, h.

- Blue:** not visited yet

- Red:** edges from nodes $\in T$ to nodes $\notin T$

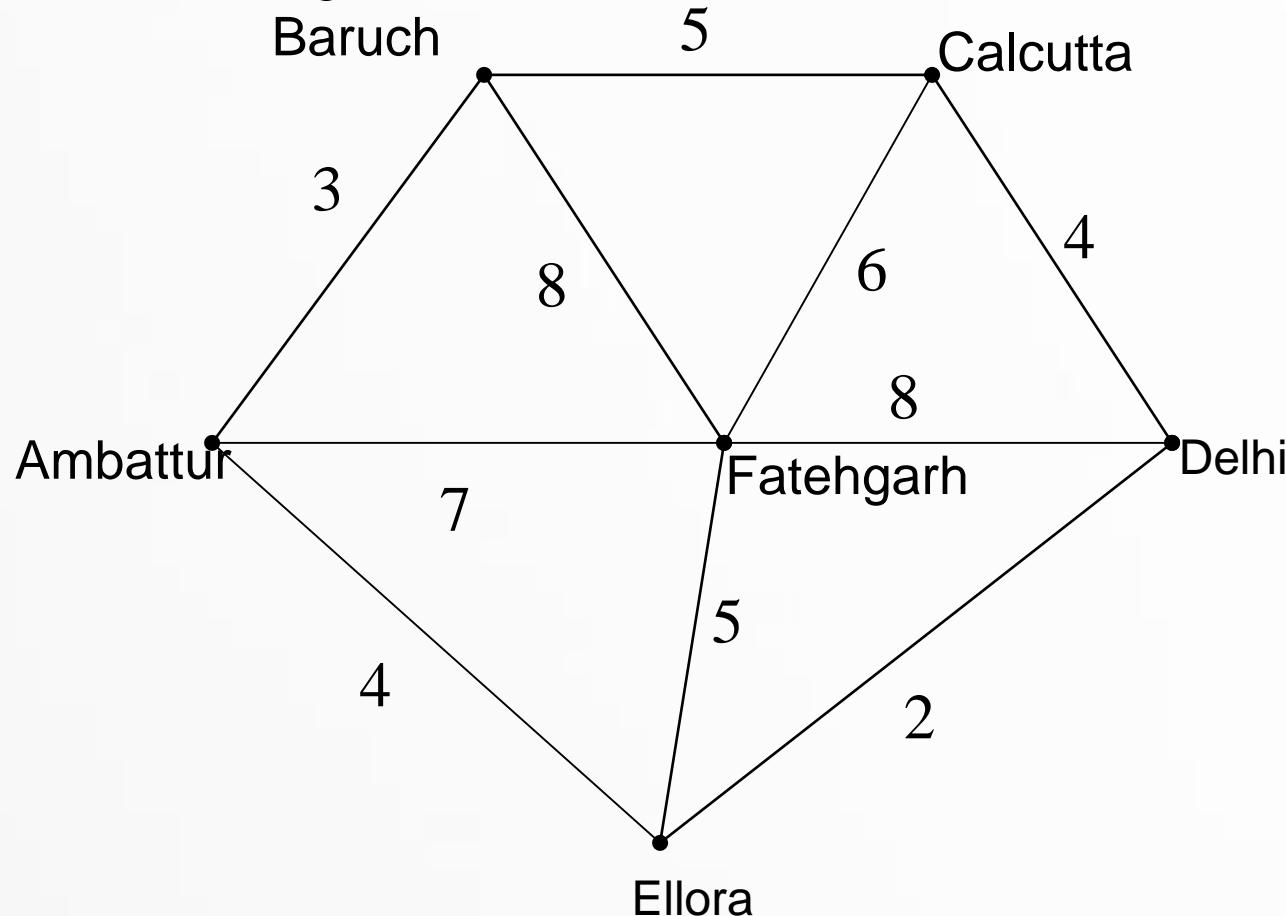
- Black:** in T

- Minimum Cost: 47



Prim's algorithm

A cable company want to connect five villages to their network which currently extends to the market town of Ambattur. What is the minimum length of cable needed?



Prim's algorithm

Prim's algorithm

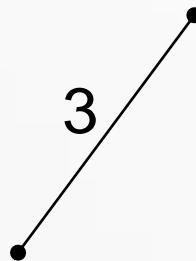
Note, this example has outgoing edges on the columns and incoming on the rows, so it is the transpose of adjacency matrix mentioned in class. Actually, it is an undirected, so $A^T = A$.

	A	B	C	D	E	F
A	-	3	-	-	4	7
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

- Start at vertex A. Label column A “1” .
- Delete row A
- Select the smallest entry in column A (AB, length 3)

Baruch

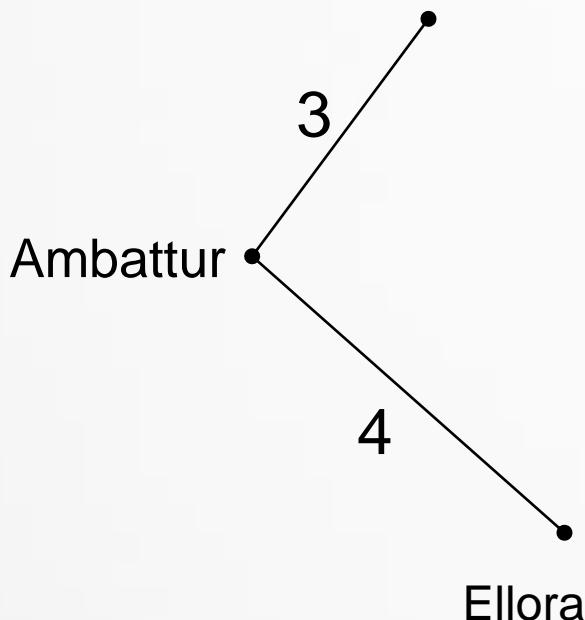


	A	B	C	D	E	F
A	-	3	-	-	4	7
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

- Label column B “2”
- Delete row B
- Select the smallest uncovered entry in either column A or column B (AE, length 4)

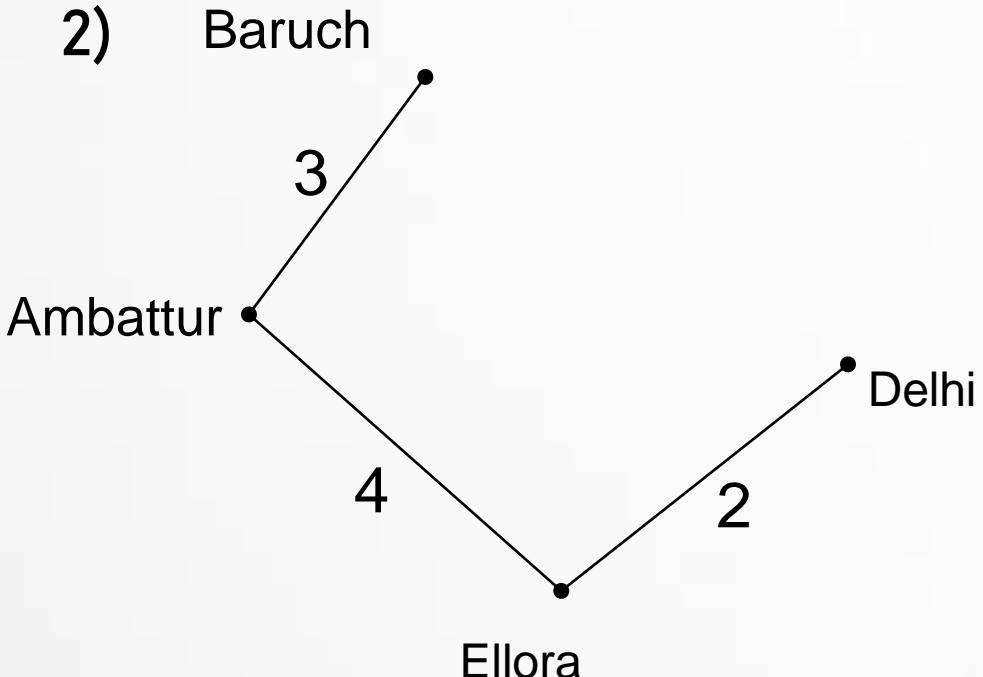
Baruch



	1	2					
	A	B	C	D	E	F	
A	-	3	-	-	4	7	
B	3	-	5	-	-	8	
C	-	5	-	4	-	6	
D	-	-	4	-	2	8	
E	4	-	-	2	-	5	
F	7	8	6	8	5	-	

Prim's algorithm

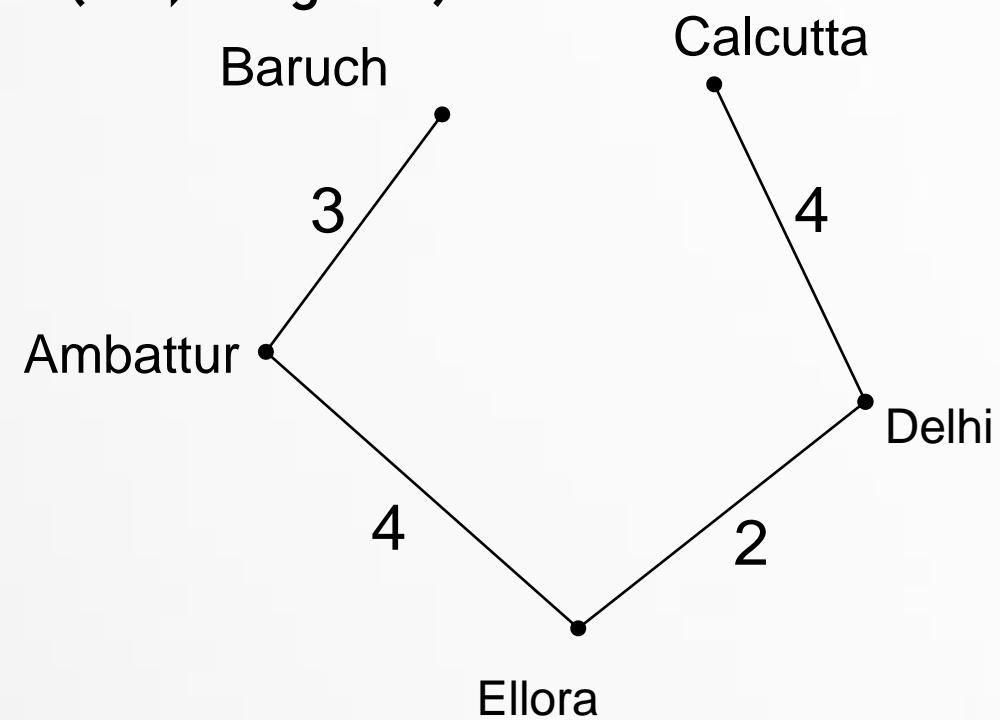
- Label column E “3”
- Delete row E
- Select the smallest uncovered entry in either column A, B or E (ED, length 2)



	1	2	3			
	A	B	C	D	E	F
A	-	3	-	-	4	7
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

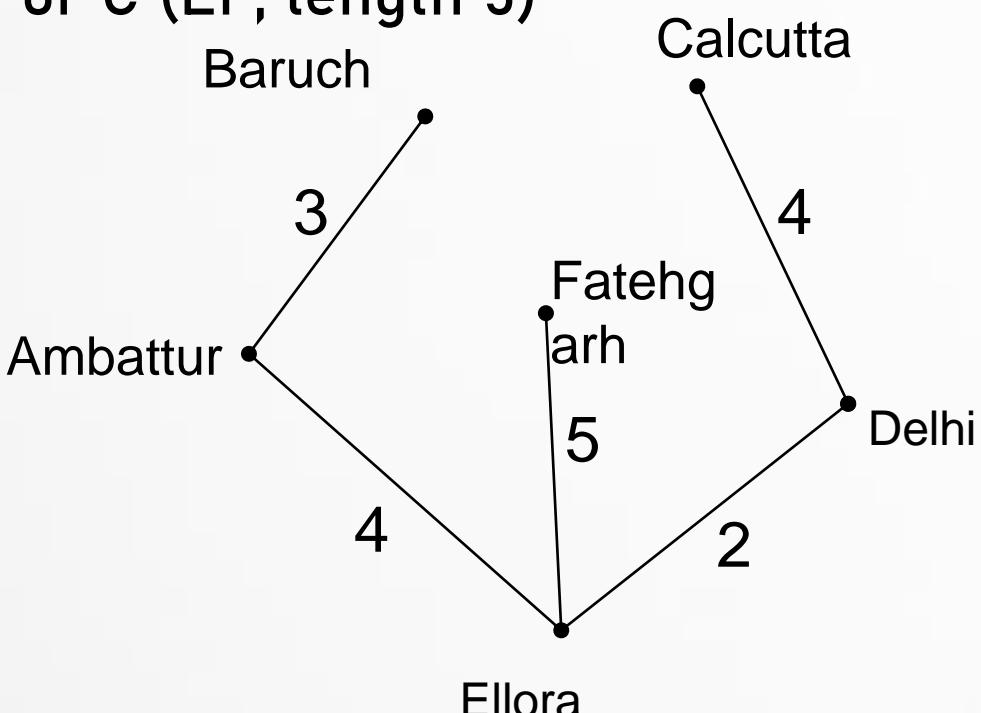
- Label column D “4”
- Delete row D
- Select the smallest uncovered entry in either column A, B, D or E (DC, length 4)



	1	2	4	3	E	F
A	-	3	-	-	4	7
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

- Label column C “5”
- Delete row C
- Select the smallest uncovered entry in either column A, B, D, E or C (EF, length 5)

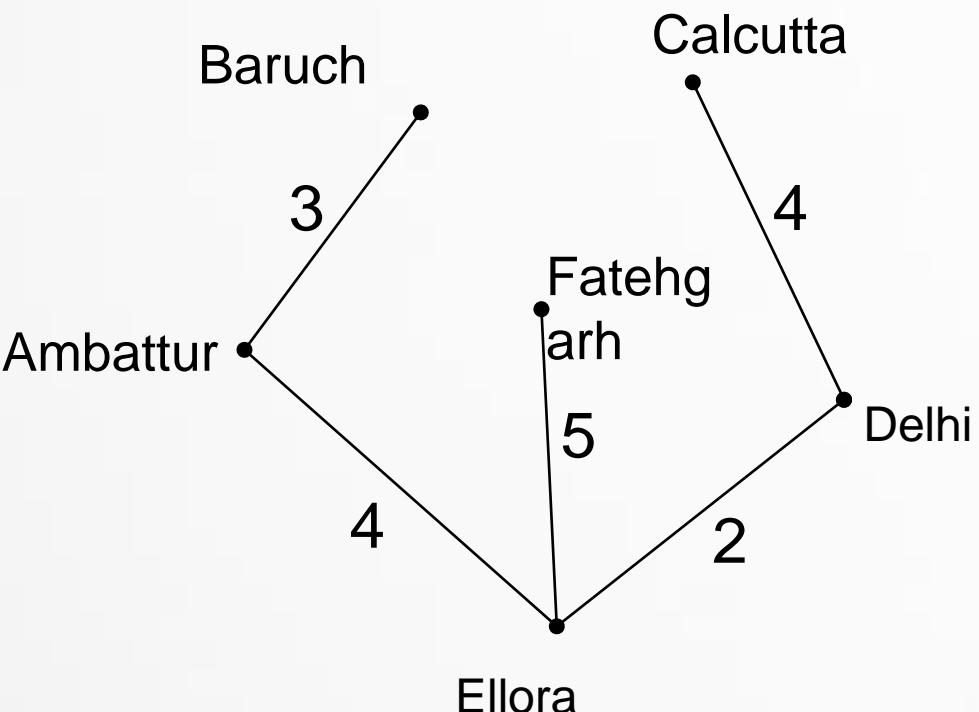


	1	2	5	4	3	F
A	-	3	-	-	4	7
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

FINALLY

- Label column F “6”
- Delete row F

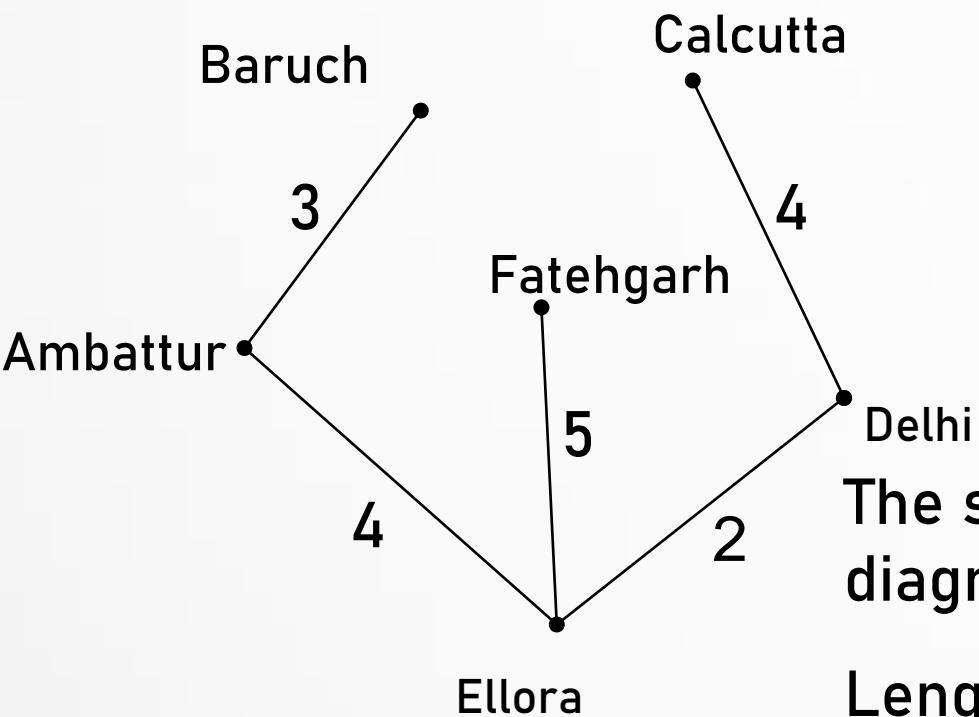


	1	2	5	4	3	6
A	-	3	-	4	7	
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

Prim's algorithm

FINALLY

- Label column F "6"
- Delete row F



	1	2	5	4	3	6
A	-	B	C	D	E	F
B	3	-	5	-	-	8
C	-	5	-	4	-	6
D	-	-	4	-	2	8
E	4	-	-	2	-	5
F	7	8	6	8	5	-

The spanning tree is shown in the diagram

$$\text{Length } 3 + 4 + 4 + 2 + 5 = 18 \text{ Km}$$

That's all for now...