

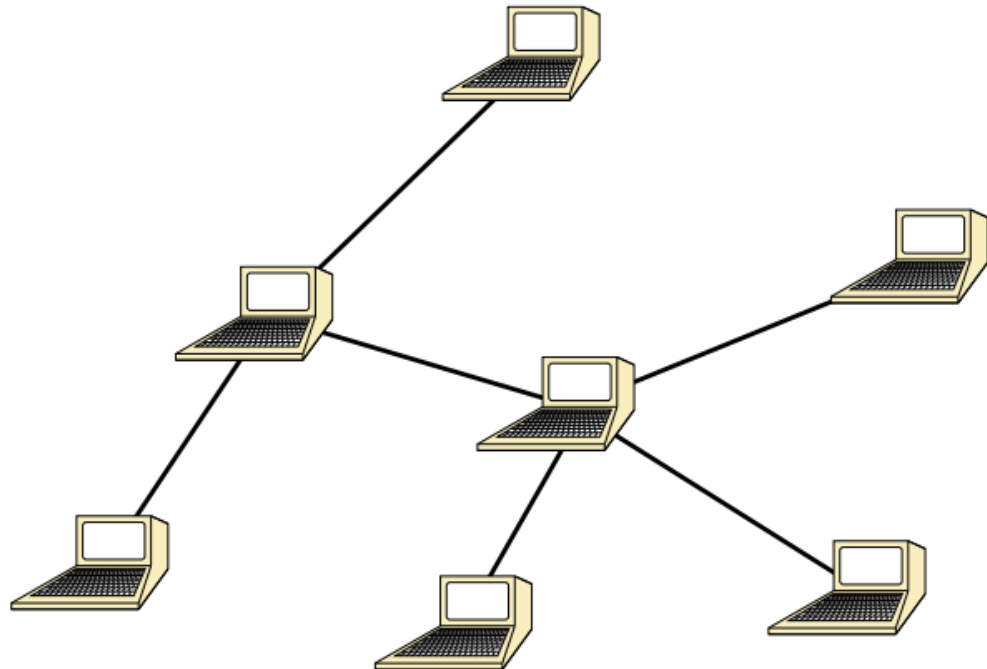


## **Chapter 23**

# **Minimum Spanning Trees**

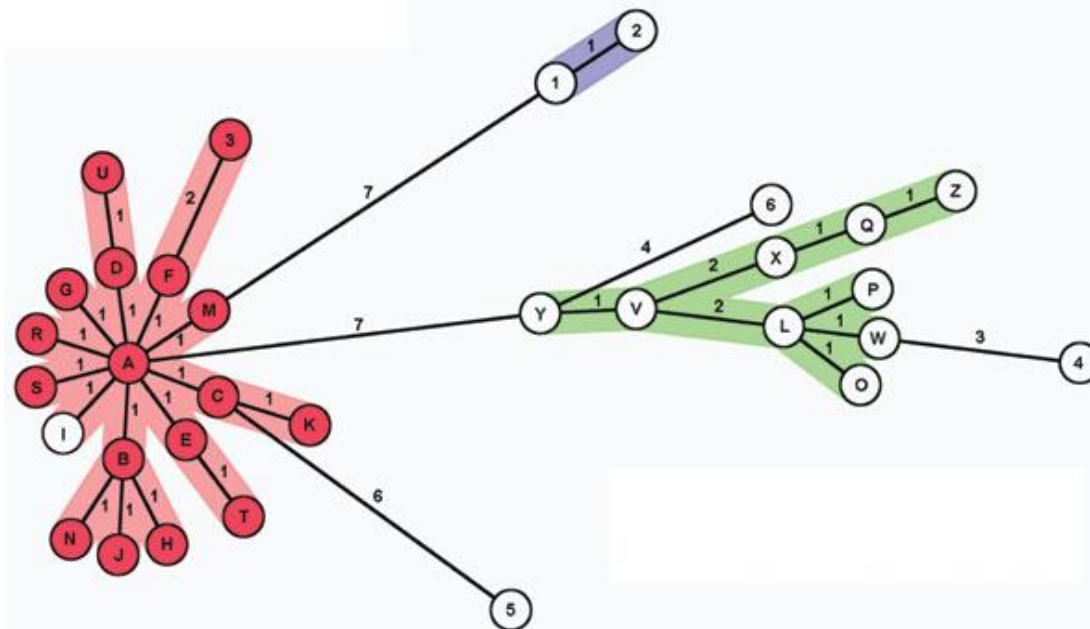
# Minimum Spanning Trees

- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Connect “nodes” with a minimum of “wire”
    - Networking
    - Circuit design



# Minimum Spanning Trees

- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Collect nearby nodes
    - Clustering, taxonomy construction



# Minimum Spanning Trees

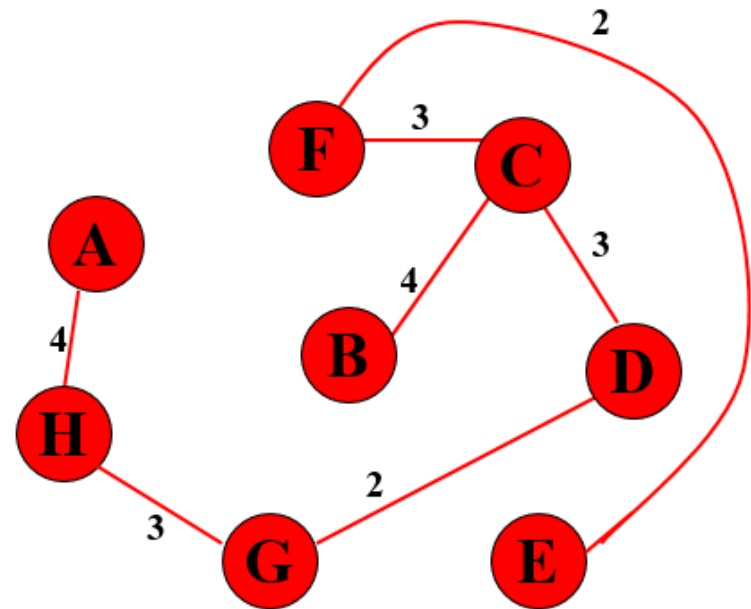
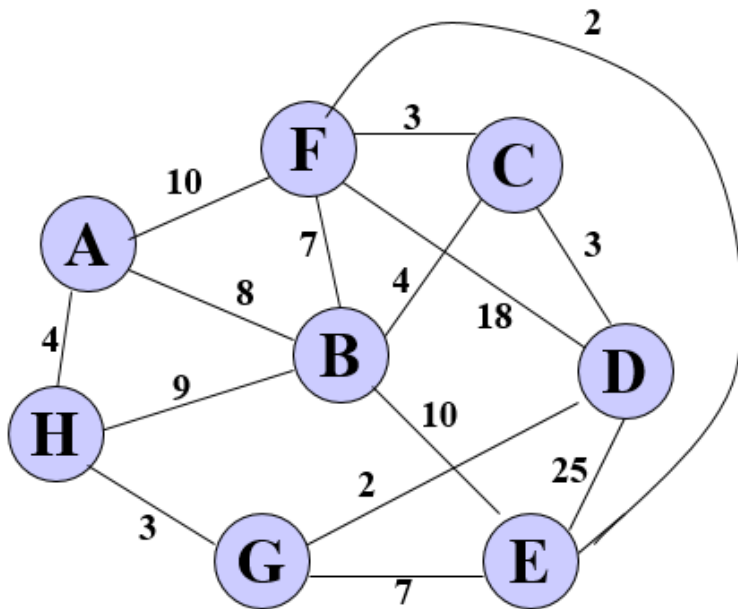
- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Approximating graphs



# Minimum Spanning Trees

- A tree is an acyclic, undirected, connected graph
- A spanning tree of a graph is a tree containing all vertices from the graph
- A minimum spanning tree is a spanning tree, where the sum of the weights on the tree's edges are minimal

# Minimum Spanning Trees



- A minimum spanning tree is a spanning tree, where the sum of the weights on the tree's edges are minimal

# Minimum Spanning Trees

- Problem formulation

- Given an undirected, weighted graph  $G = (V, E)$  with weights  $w(u, v)$  for each edge  $(u, v) \in E$
- Find an acyclic subset  $T \subseteq E$  that connects all of the vertices  $V$ , and minimizes the total weight:

$$w(T) = \sum_{(u,v) \in T} w(u, v)$$

- The minimum spanning tree is  $(V, T)$ 
  - Minimum spanning tree may be not unique (can be more than one)



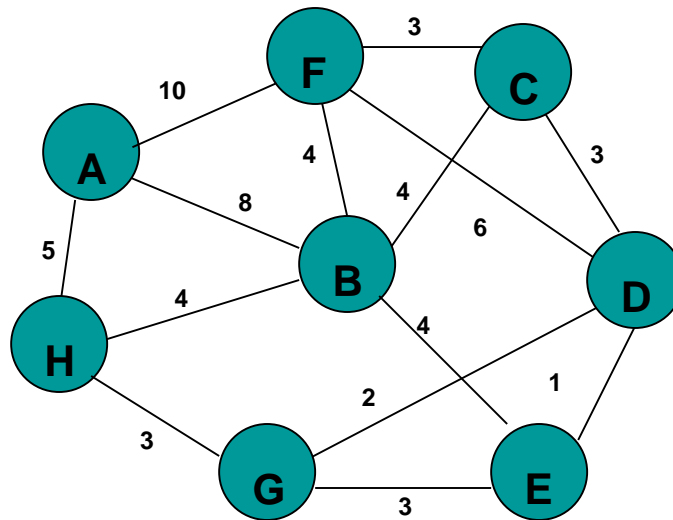
# Minimum Spanning Trees

- Both Kruskal's and Prim's Algorithms work with undirected graphs
- Both work with weighted and unweighted graphs but are more interesting when edges are weighted
- Both are greedy algorithms that produce optimal solutions



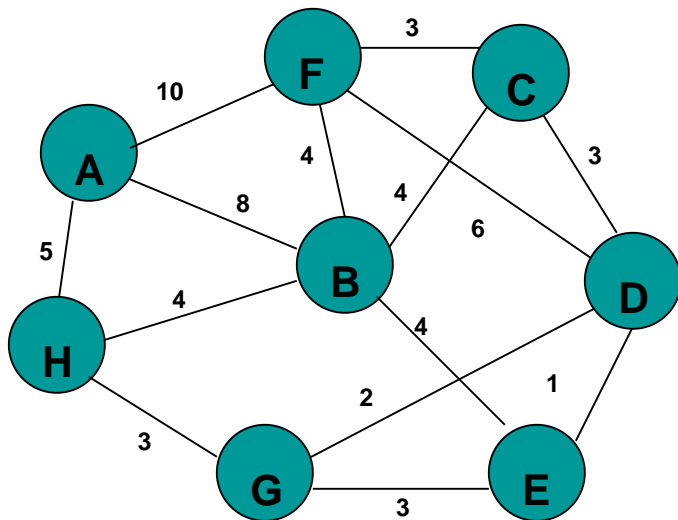
# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm
  - Work with edges
  - Two steps:
    - Sort edges by increasing edge weight
    - Select the first  $|V| - 1$  edges that do not generate a cycle
  - Walk through:



# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



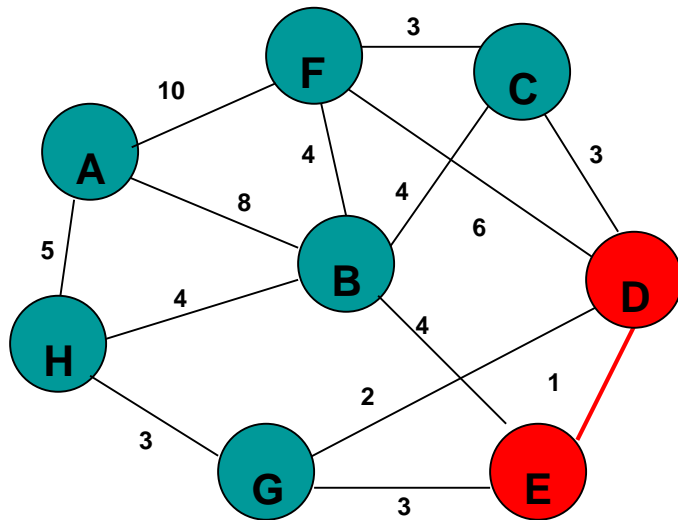
Sort the edges by increasing edge weight

<i>edge</i>	$d_v$	
(D,E)	1	
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



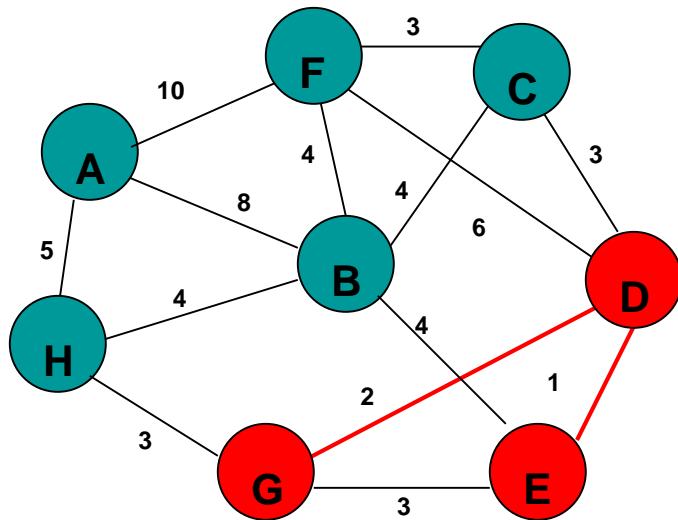
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



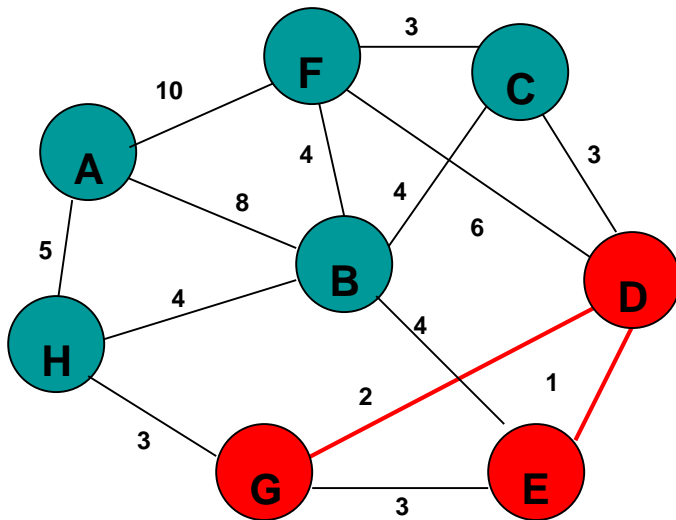
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



Select first  $|V|-1$  edges which do not generate a cycle

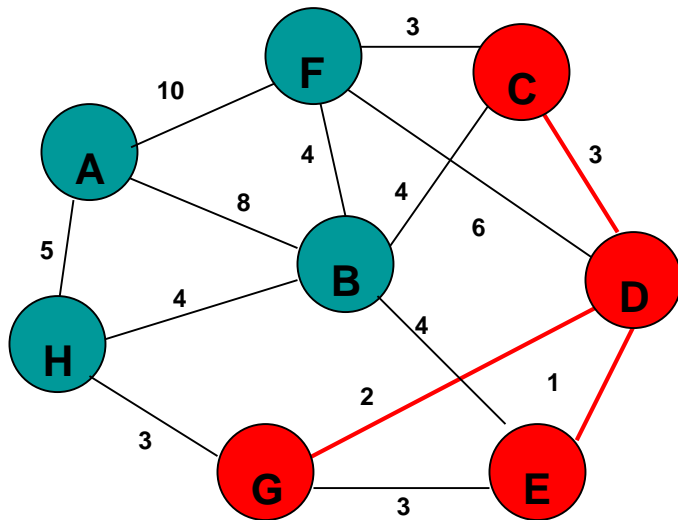
<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Accepting edge (E,G) would create a cycle

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



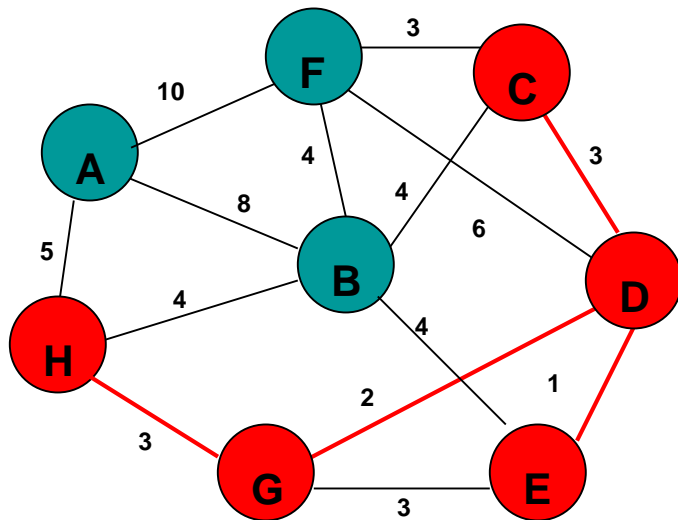
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



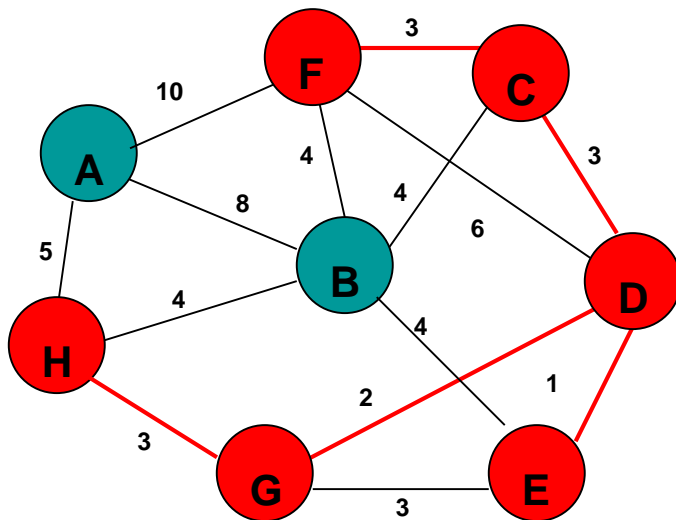
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



Select first  $|V|-1$  edges which do not generate a cycle

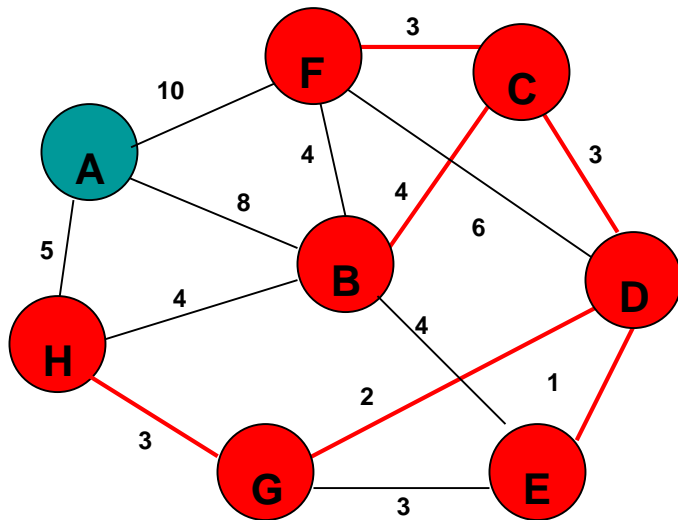
<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	



# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



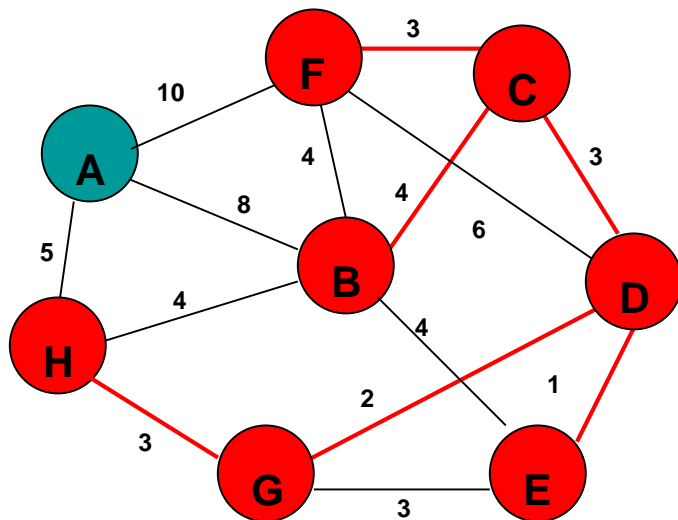
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



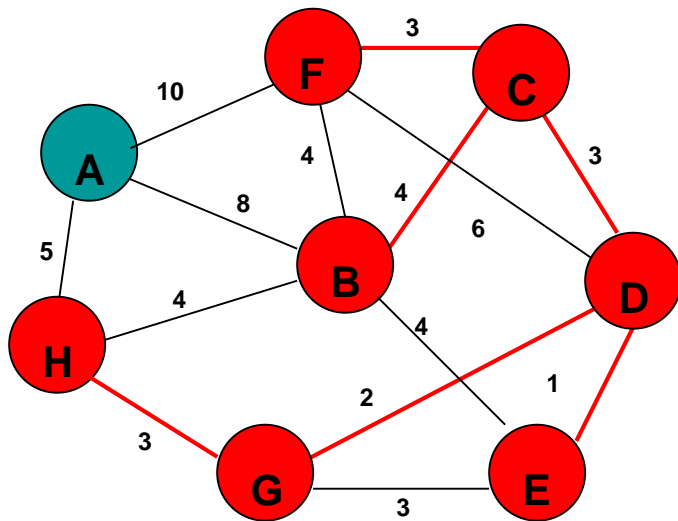
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	✗
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



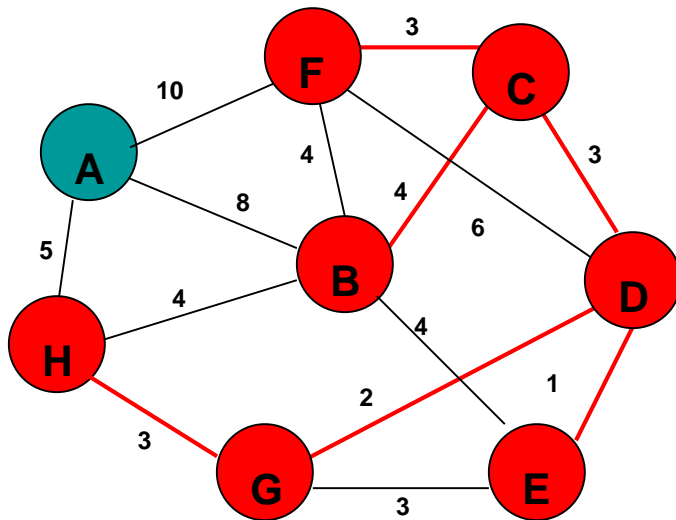
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	✗
(B,F)	4	✗
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



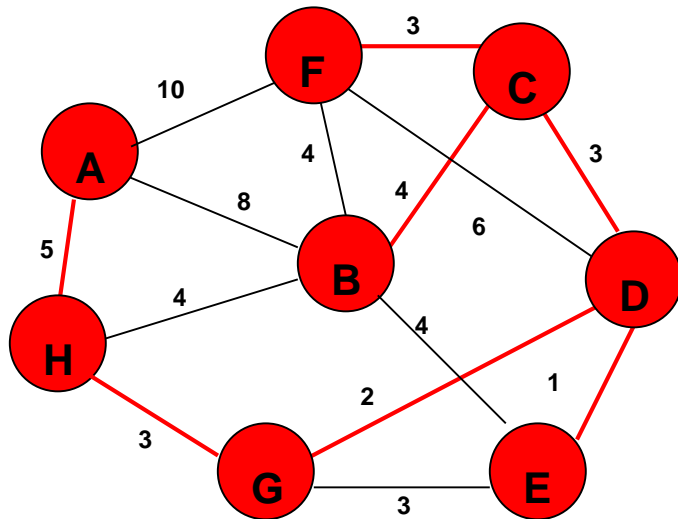
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	✗
(B,F)	4	✗
(B,H)	4	✗
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



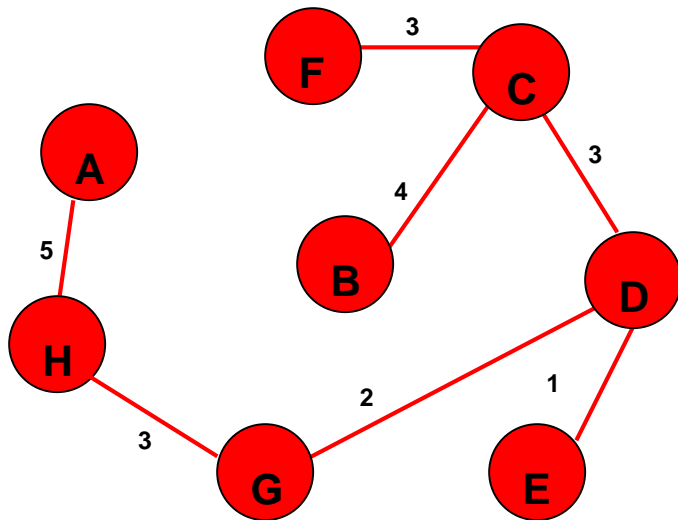
Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	✗
(B,F)	4	✗
(B,H)	4	✗
(A,H)	5	✓
(D,F)	6	
(A,B)	8	
(A,F)	10	

# Minimum Spanning Trees

- Solution 1: Kruskal's algorithm



Select first  $|V|-1$  edges which do not generate a cycle

<i>edge</i>	$d_v$	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<i>edge</i>	$d_v$	
(B,E)	4	✗
(B,F)	4	✗
(B,H)	4	✗
(A,H)	5	✓
(D,F)	6	
(A,B)	8	
(A,F)	10	

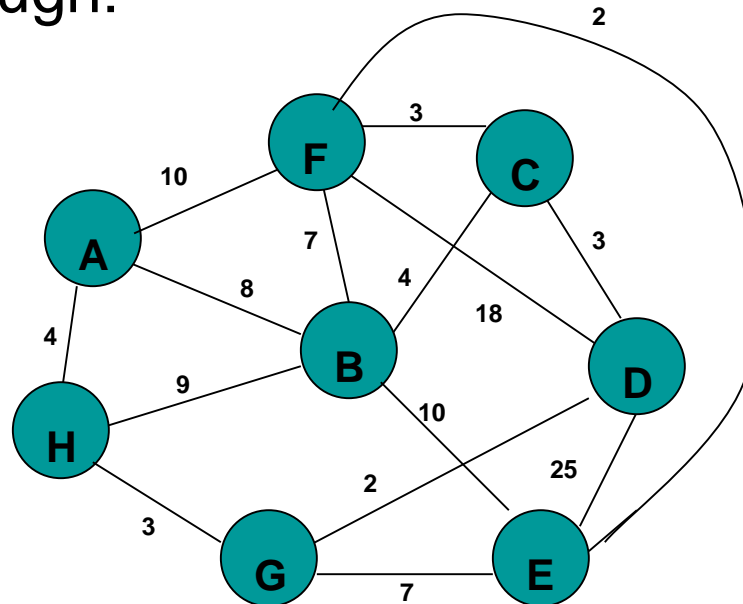
} not considered

**Done**

**Total Cost = 21**

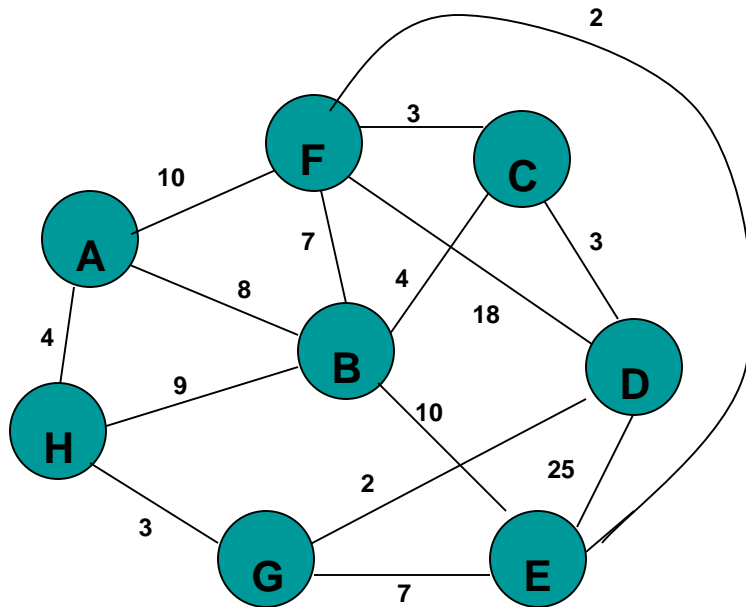
# Minimum Spanning Trees

- Solution 2: Prim's algorithm
  - Work with nodes (instead of edges)
  - Two steps
    - Select node with minimum distance
    - Update distances of adjacent, unselected nodes
  - Walk through:



# Minimum Spanning Trees

- Solution 2: Prim's algorithm



Initialize array

	$K$	$d_v$	$p_v$
<b>A</b>	F	$\infty$	—
<b>B</b>	F	$\infty$	—
<b>C</b>	F	$\infty$	—
<b>D</b>	F	$\infty$	—
<b>E</b>	F	$\infty$	—
<b>F</b>	F	$\infty$	—
<b>G</b>	F	$\infty$	—
<b>H</b>	F	$\infty$	—

$K$  : whether in the tree

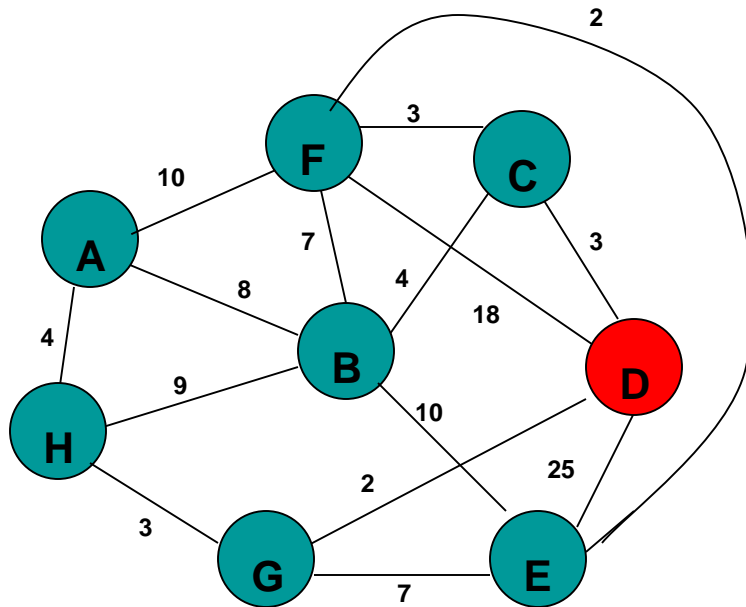
$d_v$  : distance to the tree

$p_v$  : closest node that is in the tree



# Minimum Spanning Trees

- Solution 2: Prim's algorithm

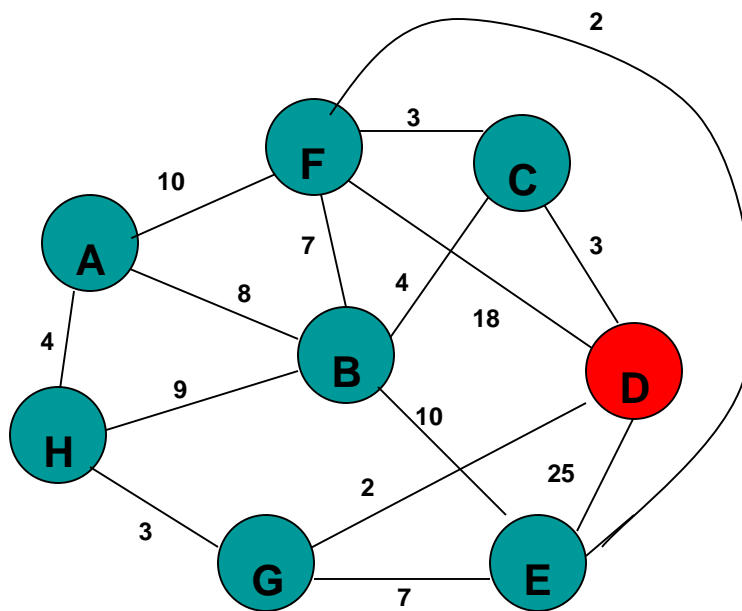


Start with any node, say D

	$K$	$d_v$	$p_v$
A			
B			
C			
D	T	0	-
E			
F			
G			
H			

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

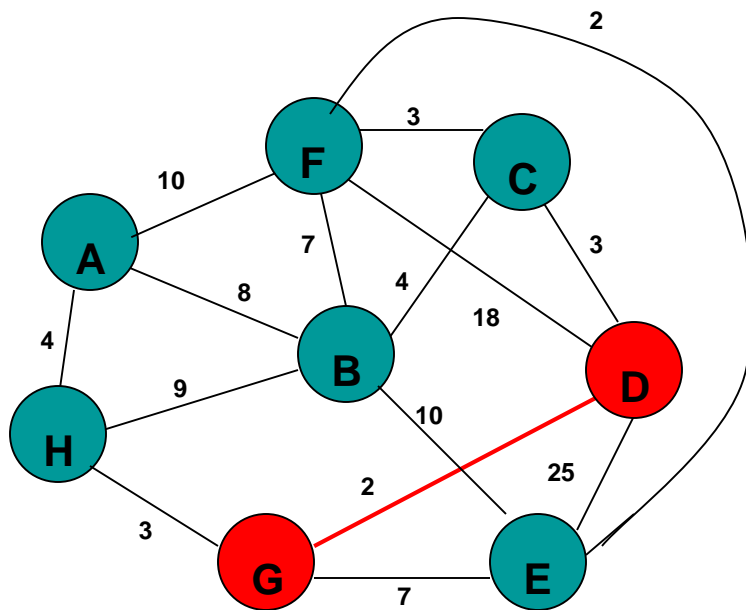


Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G		2	D
H			

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

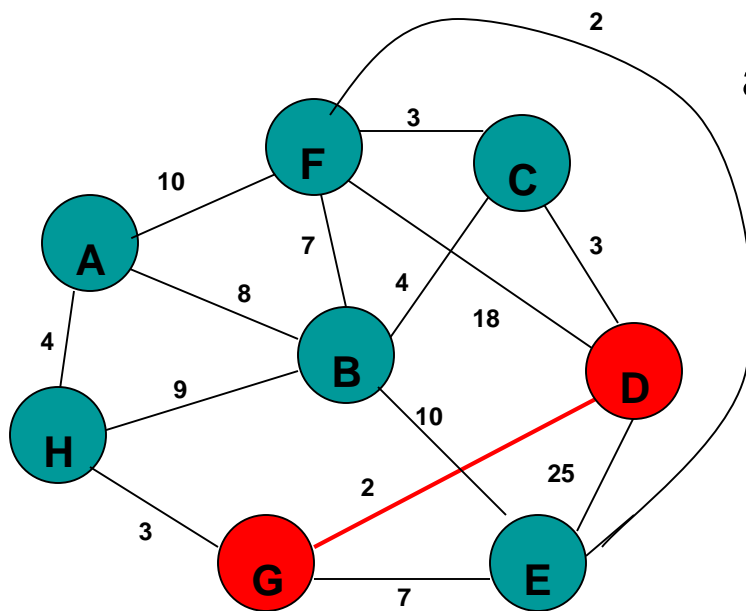


Select node with minimum distance

	$K$	$d_v$	$p_v$
A			
B			
C		3	D
D	T	0	–
E		25	D
F		18	D
G	T	2	D
H			

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

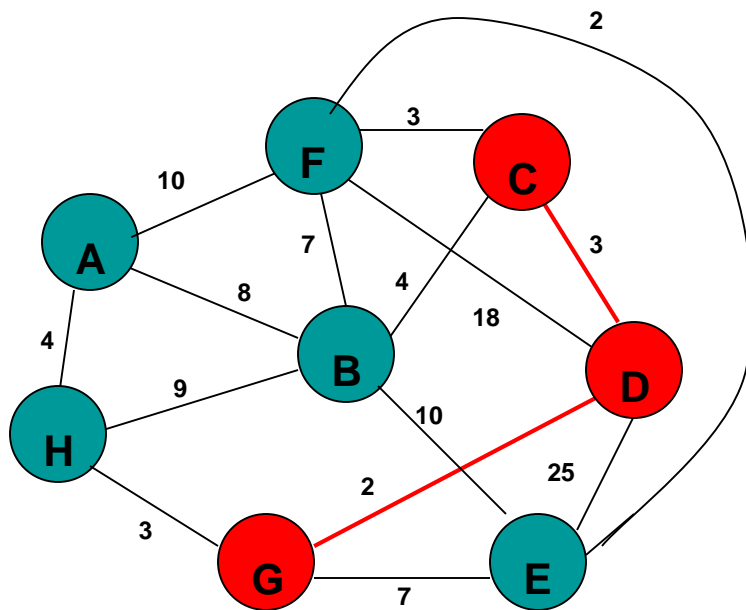


Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
A			
B			
C		3	D
D	T	0	–
E		7	G
F		18	D
G	T	2	D
H		3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

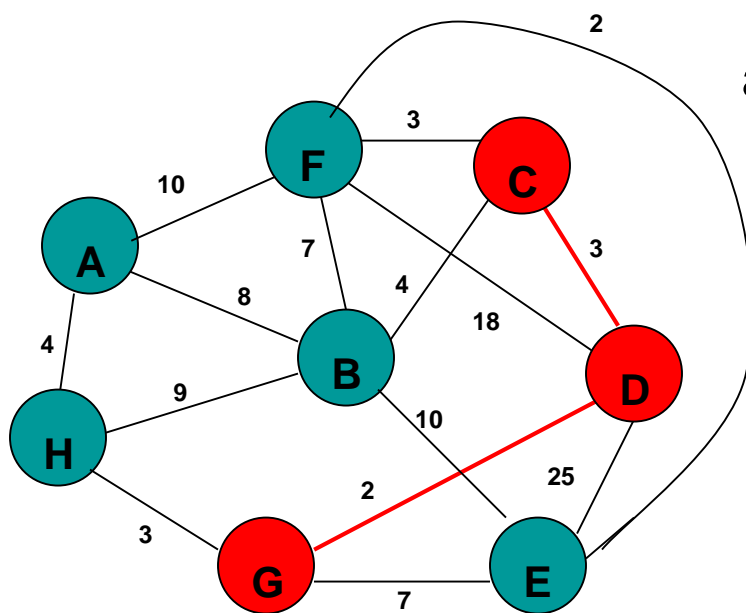


Select node with minimum distance

	$K$	$d_v$	$p_v$
<b>A</b>			
<b>B</b>			
<b>C</b>	<b>T</b>	3	D
<b>D</b>	T	0	–
<b>E</b>		7	G
<b>F</b>		18	D
<b>G</b>	T	2	D
<b>H</b>		3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

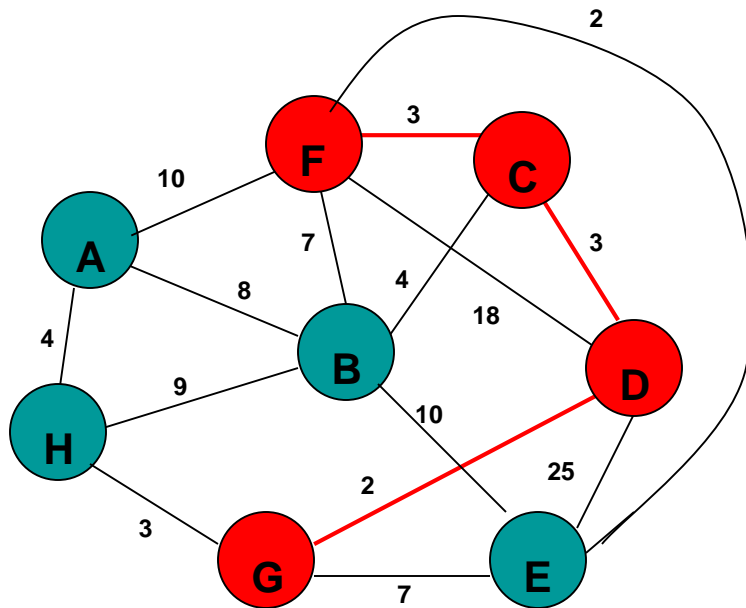


Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
A			
B		4	C
C	T	3	D
D	T	0	–
E		7	G
F		3	C
G	T	2	D
H		3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

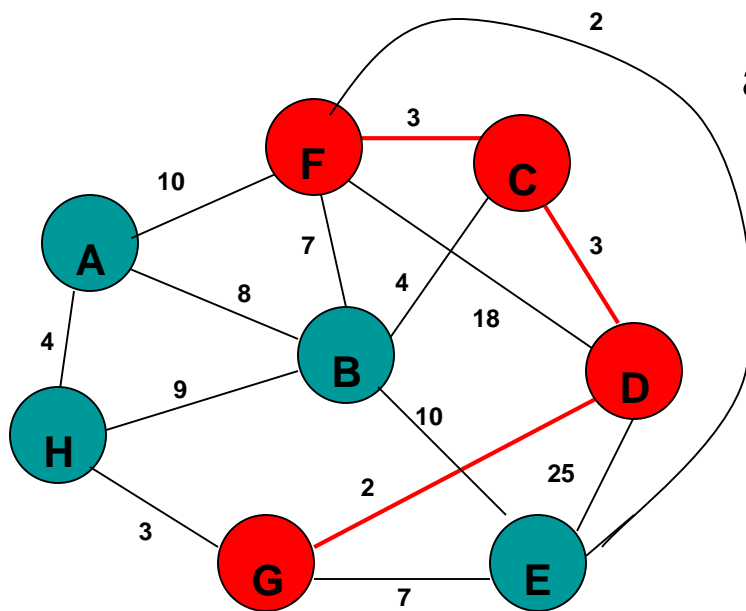


Select node with minimum distance

	$K$	$d_v$	$p_v$
<b>A</b>			
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>		7	G
<b>F</b>	<b>T</b>	3	C
<b>G</b>	T	2	D
<b>H</b>		3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm



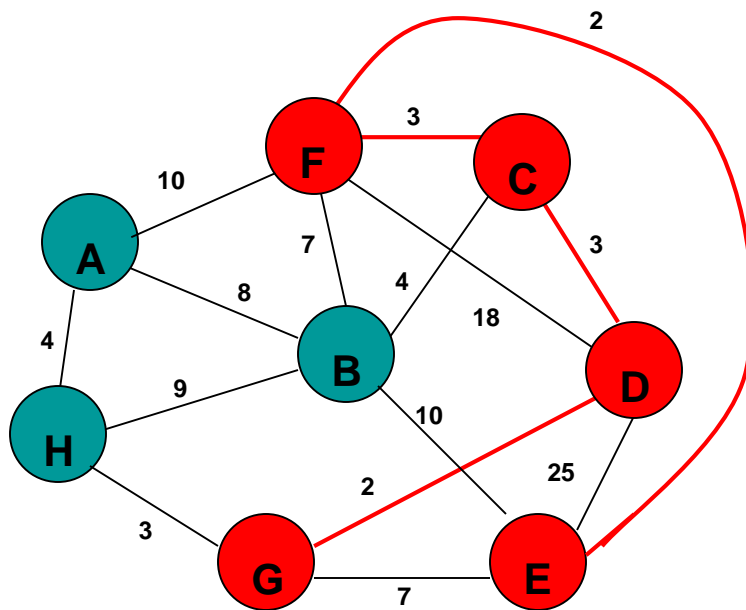
Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
<b>A</b>		10	F
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>		2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>		3	G



# Minimum Spanning Trees

- Solution 2: Prim's algorithm

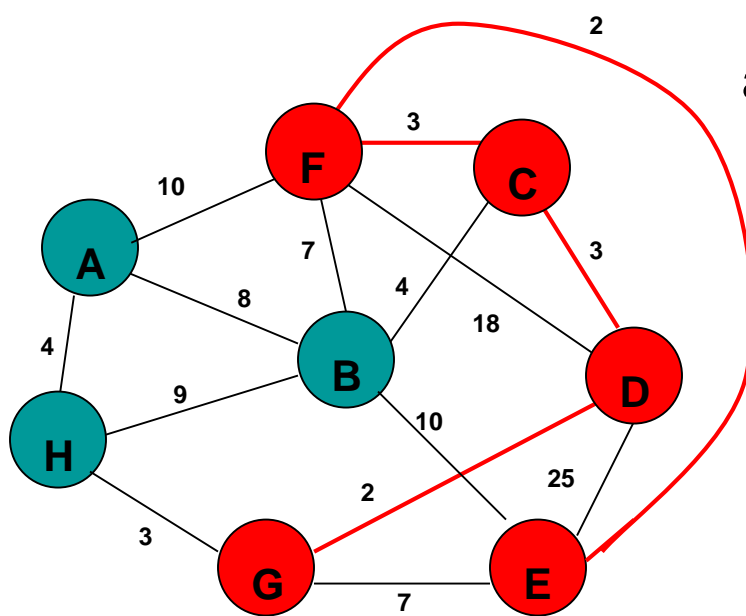


Select node with minimum distance

	$K$	$d_v$	$p_v$
<b>A</b>		10	F
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	<b>T</b>	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>		3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm



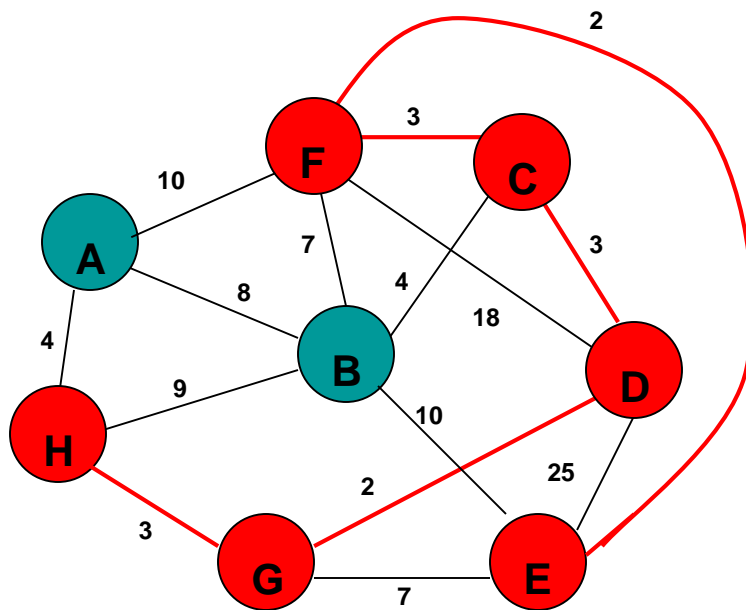
Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
<b>A</b>		10	F
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>		3	G

Table entries unchanged

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

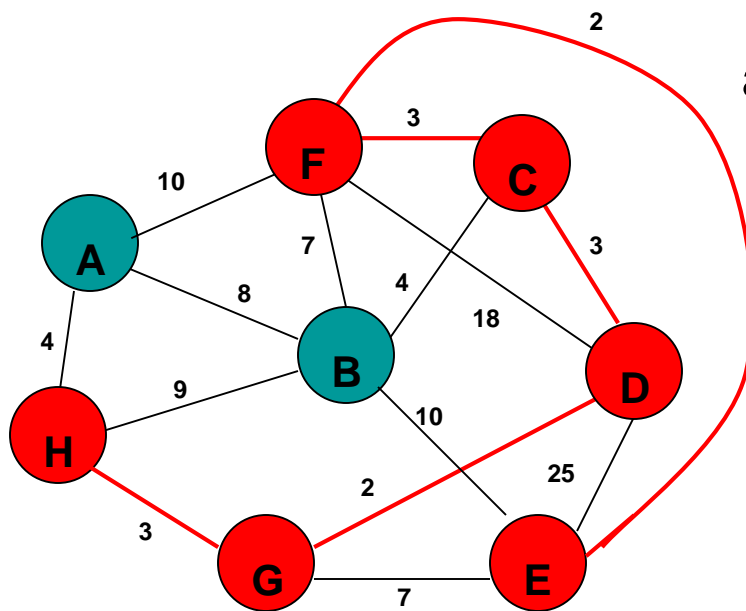


Select node with minimum distance

	$K$	$d_v$	$p_v$
<b>A</b>		10	F
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	<b>T</b>	3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

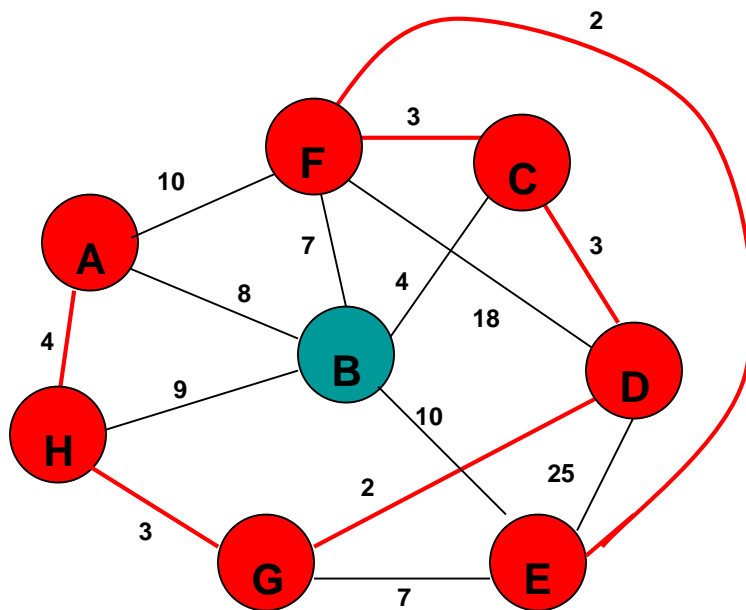


Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
<b>A</b>		4	H
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	T	3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

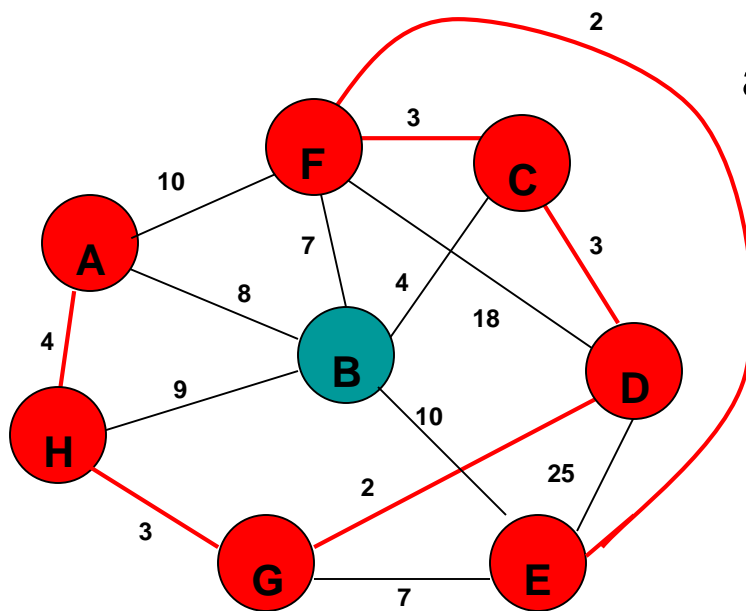


Select node with minimum distance

	<i>K</i>	$d_v$	$p_v$
<b>A</b>	<b>T</b>	4	H
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	T	3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm



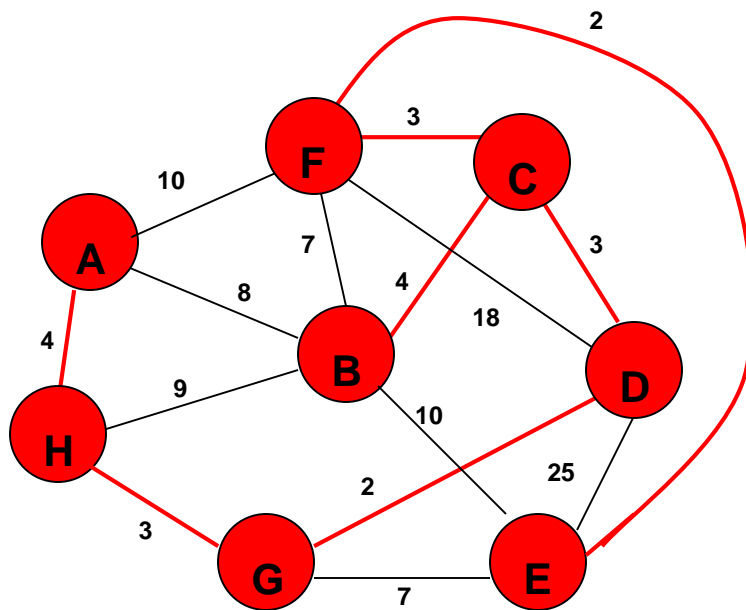
Update distances of adjacent, unselected nodes

	$K$	$d_v$	$p_v$
<b>A</b>	T	4	H
<b>B</b>		4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	T	3	G

Table entries unchanged

# Minimum Spanning Trees

- Solution 2: Prim's algorithm

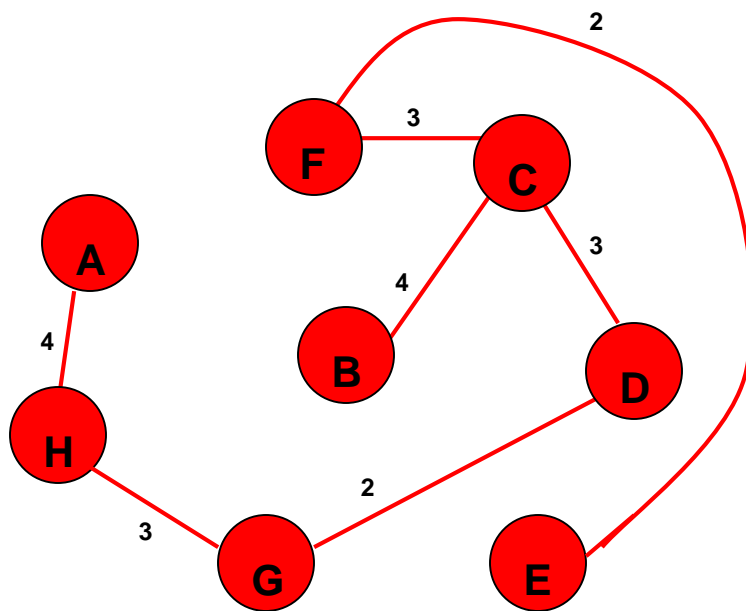


Select node with minimum distance

	$K$	$d_v$	$p_v$
<b>A</b>	T	4	H
<b>B</b>	<b>T</b>	4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	T	3	G

# Minimum Spanning Trees

- Solution 2: Prim's algorithm



Cost of Minimum Spanning Tree = **21**

	$K$	$d_v$	$p_v$
<b>A</b>	T	4	H
<b>B</b>	T	4	C
<b>C</b>	T	3	D
<b>D</b>	T	0	–
<b>E</b>	T	2	F
<b>F</b>	T	3	C
<b>G</b>	T	2	D
<b>H</b>	T	3	G

**Done**



# Minimum Spanning Trees

- Runtime

- When using binary heaps, the runtime of the Kruskal's algorithm is  $O(E \lg V)$
- When using binary heaps, the runtime of the Prim's algorithm is  $O(E \lg V)$   
When using the Fibonacci heaps, the runtime of the Prim's algorithm becomes:  $O(E + V \lg V)$
- So, when an undirected graph is dense (i.e.,  $|V|$  is much smaller than  $|E|$ ), the Prim's algorithm is more efficient