

MINIMUM SPANNING TREES

Prim's Algorithm

- Similar to Dijkstra's Algorithm

Kruskal's Algorithm

- Focuses on edges, rather than nodes



DEFINITION

- A Minimum Spanning Tree (MST) is a subgraph of an undirected graph such that the subgraph spans (includes) all nodes, is connected, is acyclic, and has minimum total edge weight



ALGORITHM CHARACTERISTICS

- Both Prim's and Kruskal'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

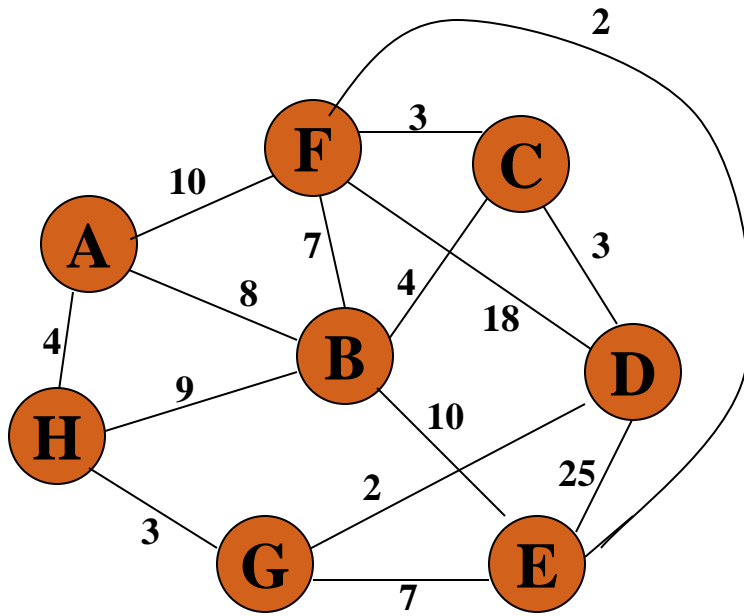


PRIM'S ALGORITHM

- Similar to Dijkstra's Algorithm except that d_v records edge weights, not path lengths



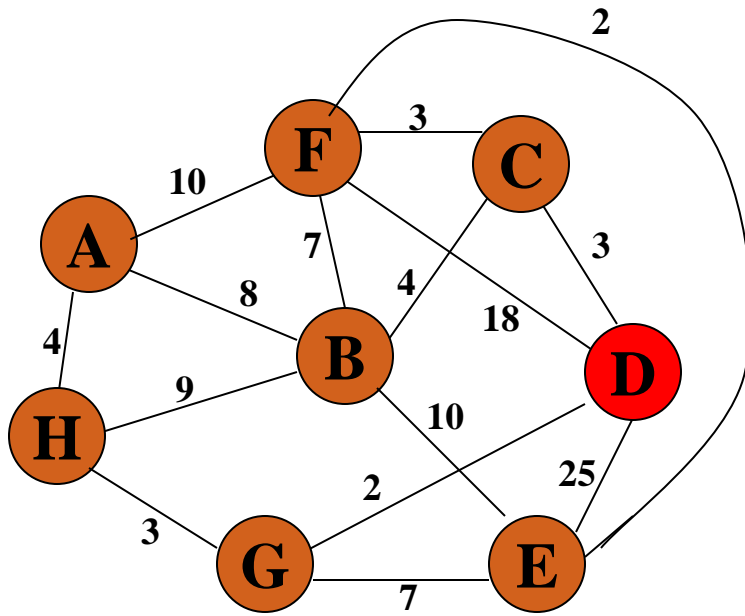
WALK-THROUGH



Initialize array

	K	d_v	p_v
A	F	∞	—
B	F	∞	—
C	F	∞	—
D	F	∞	—
E	F	∞	—
F	F	∞	—
G	F	∞	—
H	F	∞	—

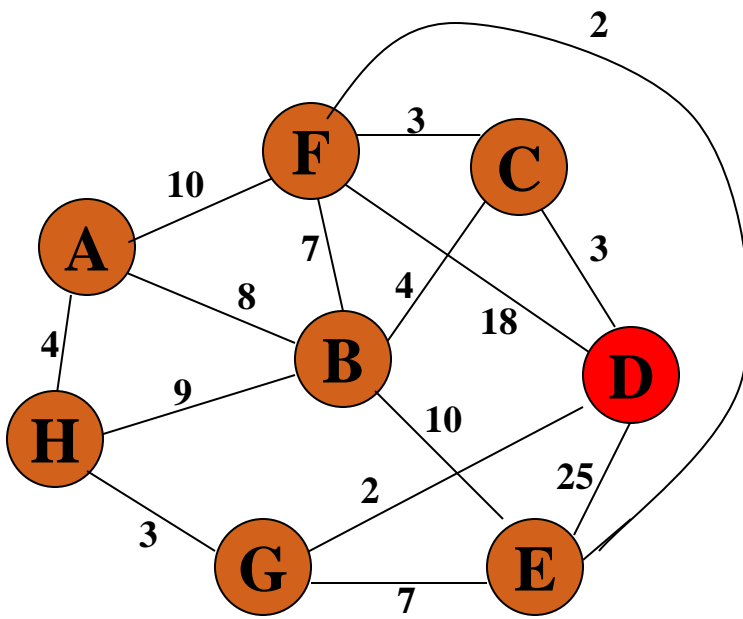




Start with any node, say D

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

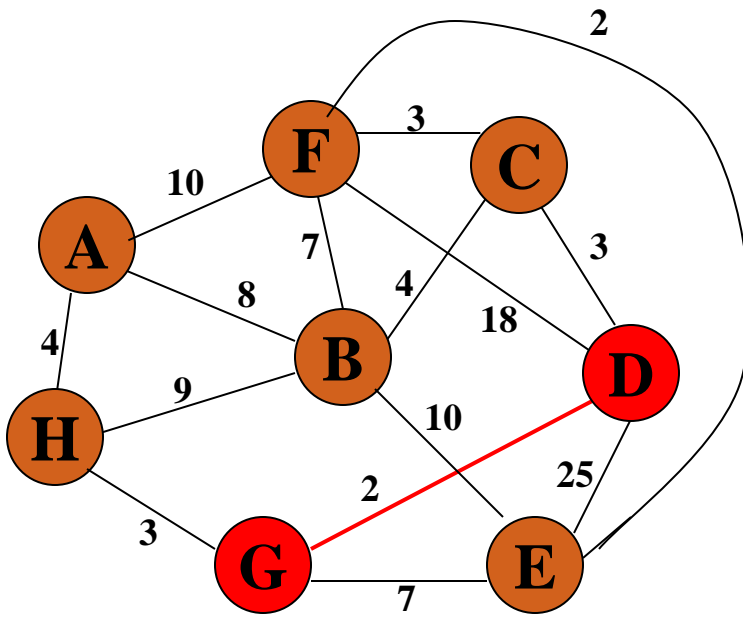




Update distances of adjacent, unselected nodes

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

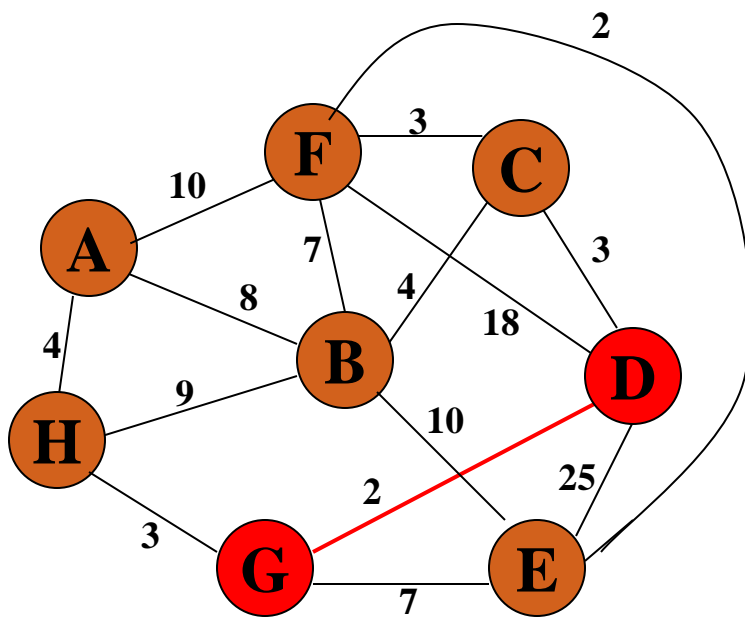




Select node with minimum distance

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

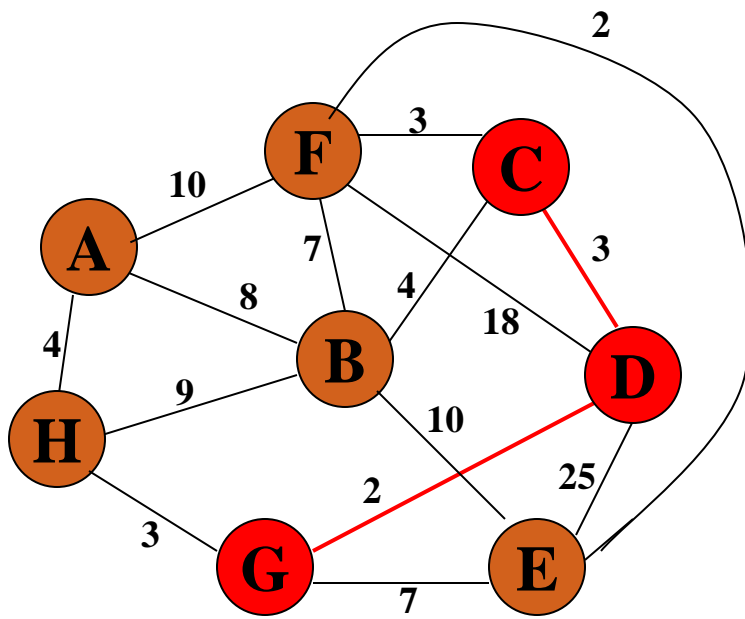




Update distances of adjacent, unselected nodes

	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

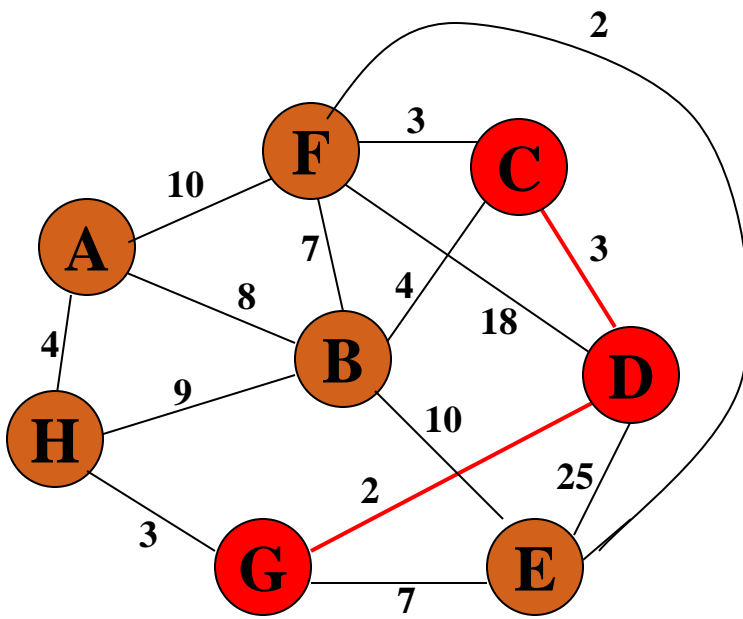




Select node with minimum distance

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

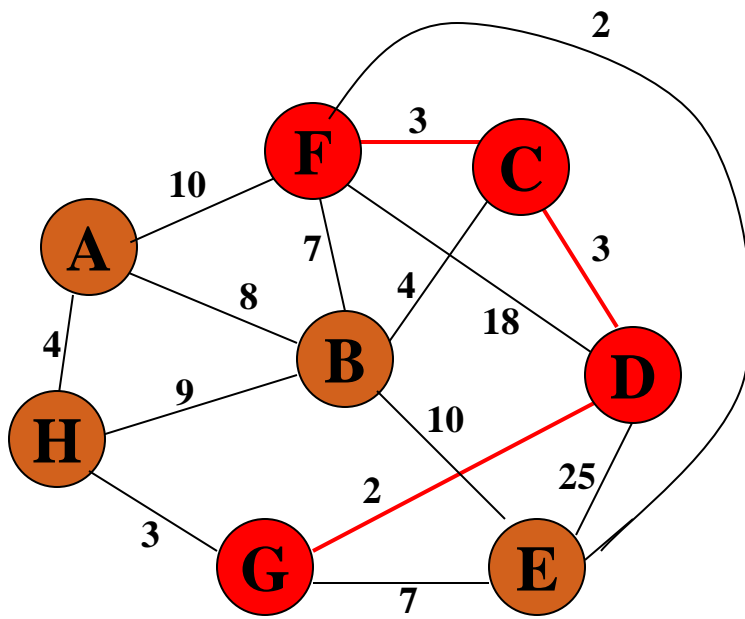




Update distances of adjacent, unselected nodes

	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

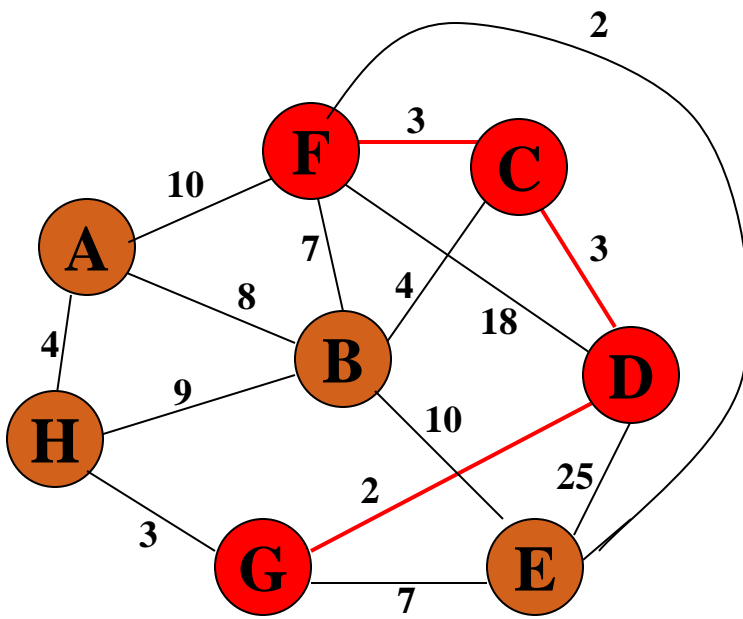




Select node with minimum distance

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

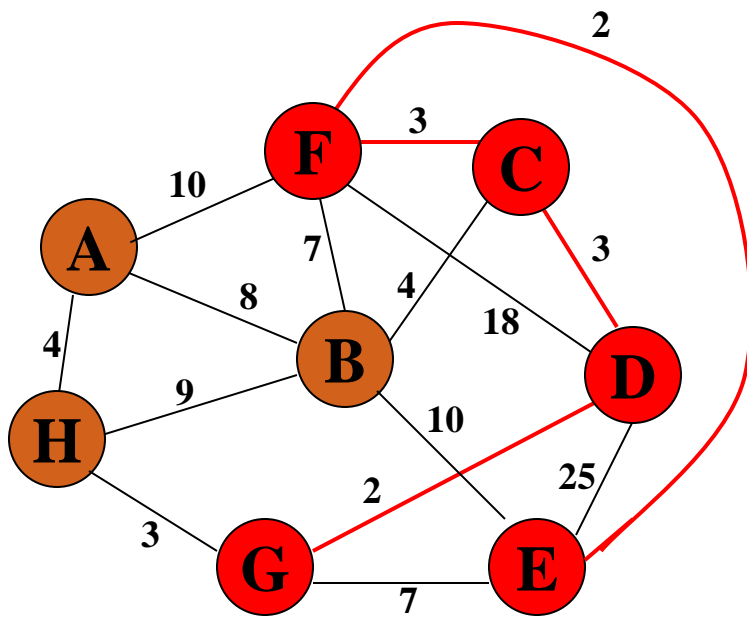




Update distances of adjacent, unselected nodes

	K	d_v	p_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E		2	F
F	T	3	C
G	T	2	D
H		3	G

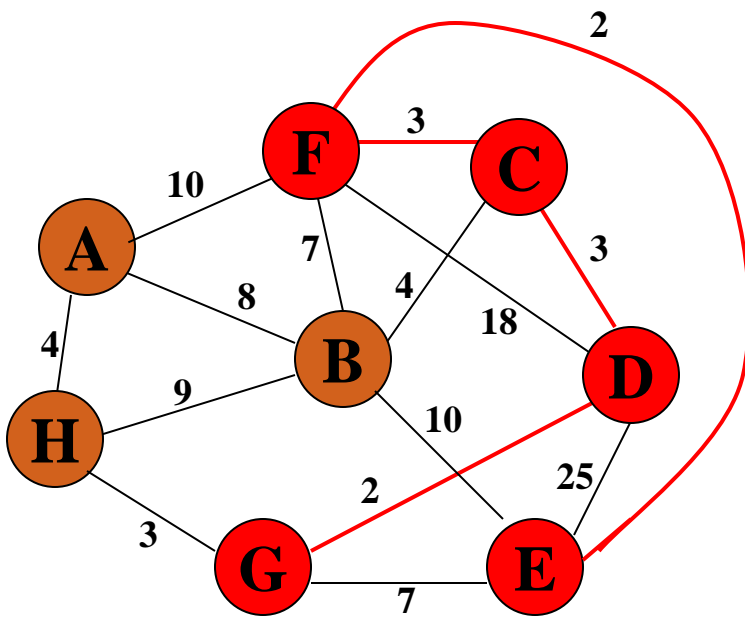




Select node with minimum distance

	K	d_v	p_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G



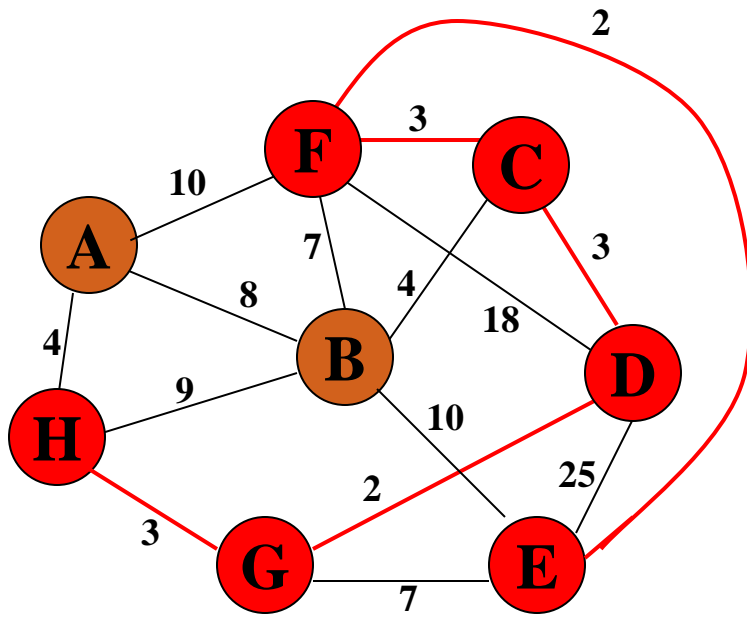


Update distances of adjacent, unselected nodes

	K	d_v	p_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H		3	G

Table entries unchanged

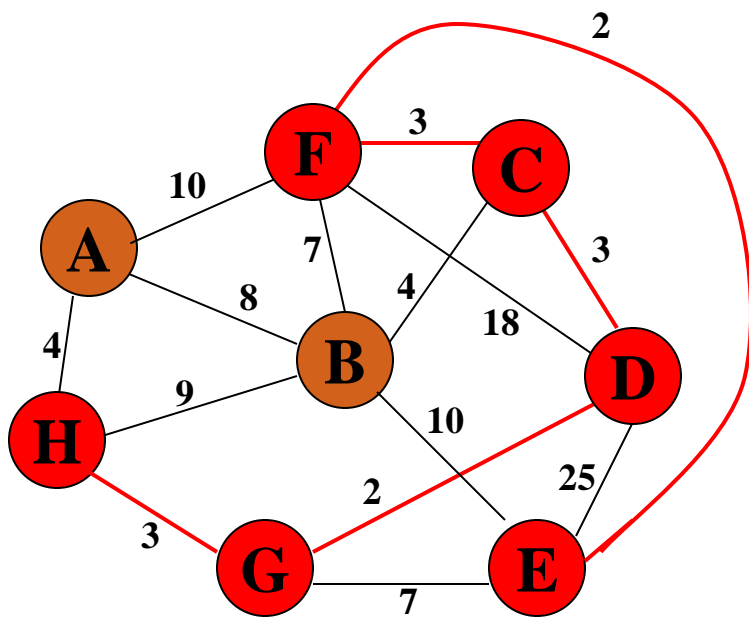




Select node with minimum distance

	K	d_v	p_v
A		10	F
B		4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

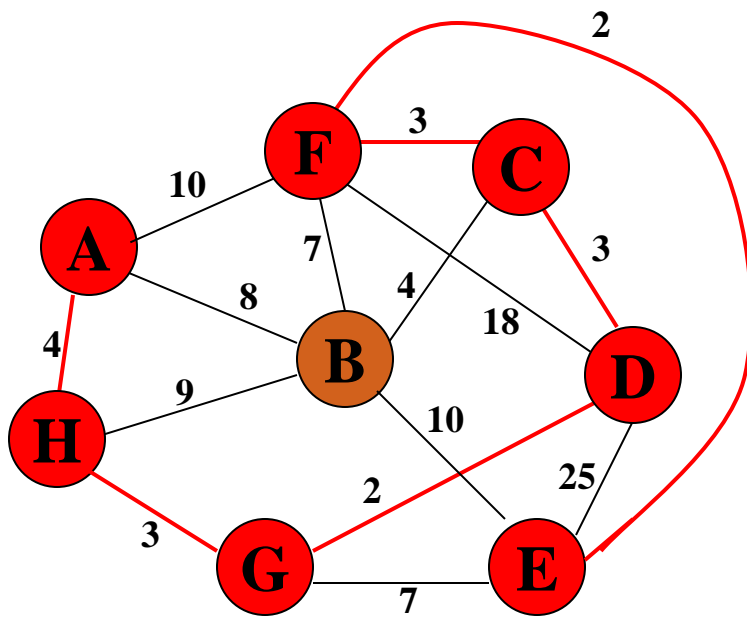




Update distances of adjacent, unselected nodes

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

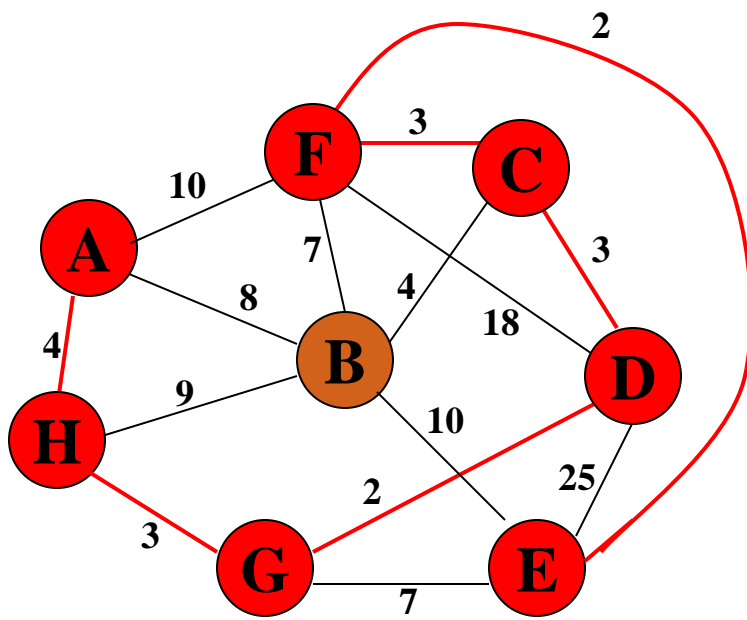




Select node with minimum distance

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



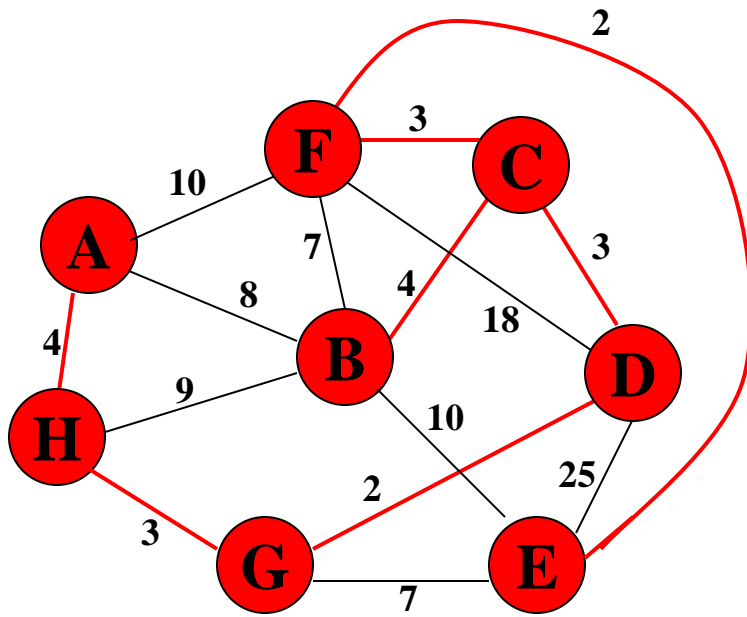


Update distances of adjacent, unselected nodes

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

Table entries unchanged

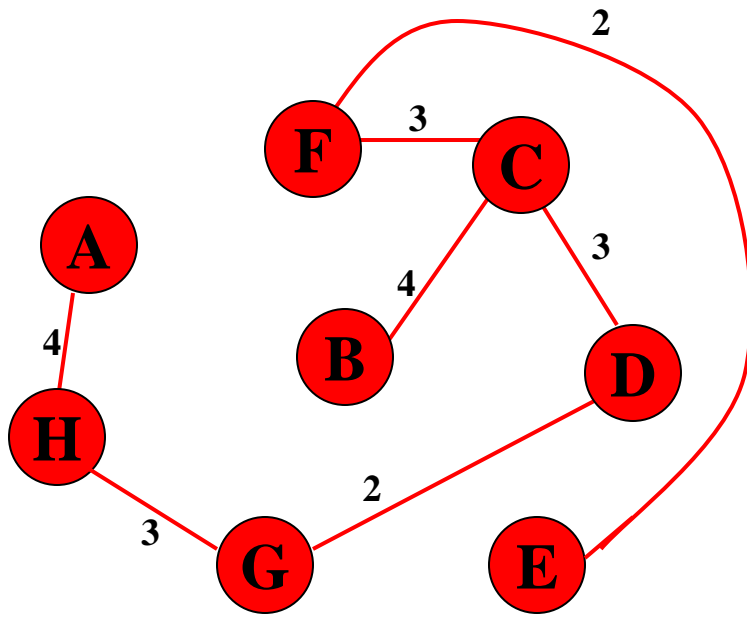




Select node with minimum distance

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





Cost of Minimum
Spanning Tree = $\sum d_v = 21$

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

Done



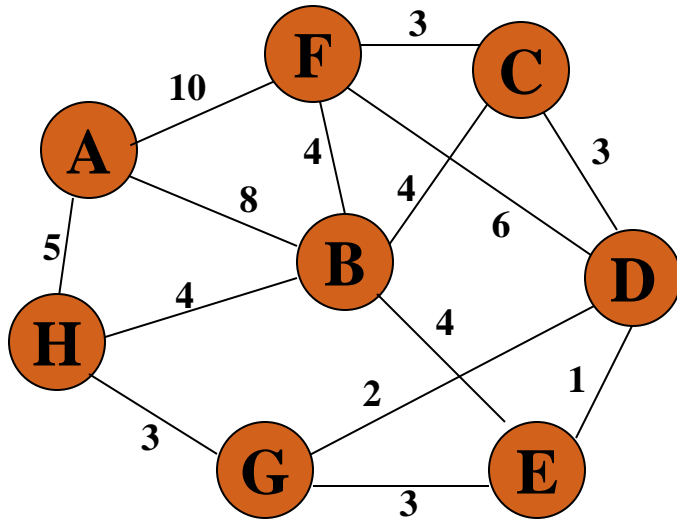
Kruskal's Algorithm

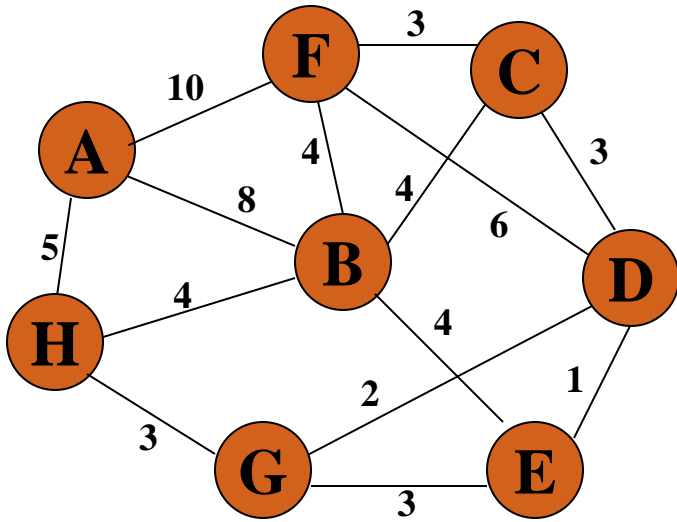
- Work with edges, rather than nodes
- Two steps:
 - Sort edges by increasing edge weight
 - Select the first $|V| - 1$ edges that do not generate a cycle



Walk-Through

Consider an undirected, weight graph





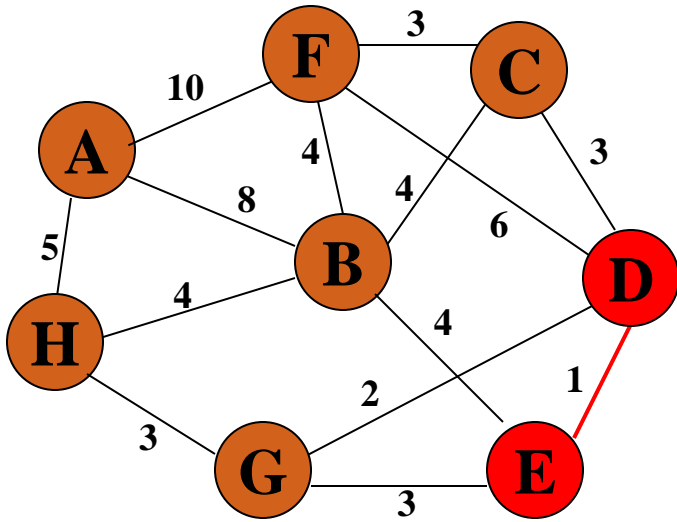
Sort the edges by increasing edge weight

<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	



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

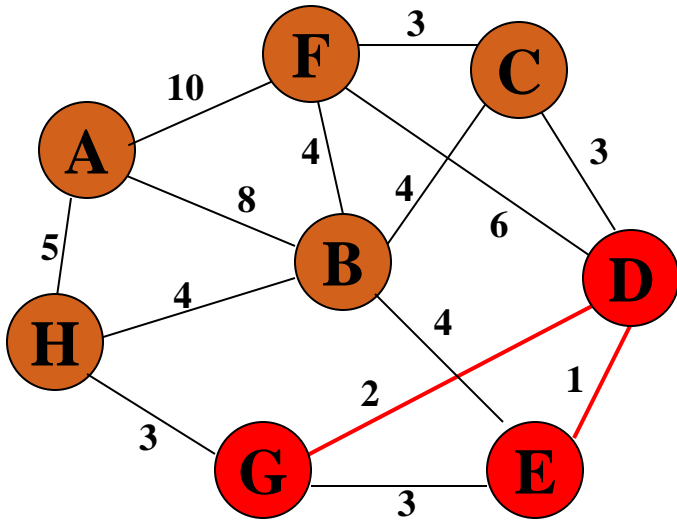


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<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	



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

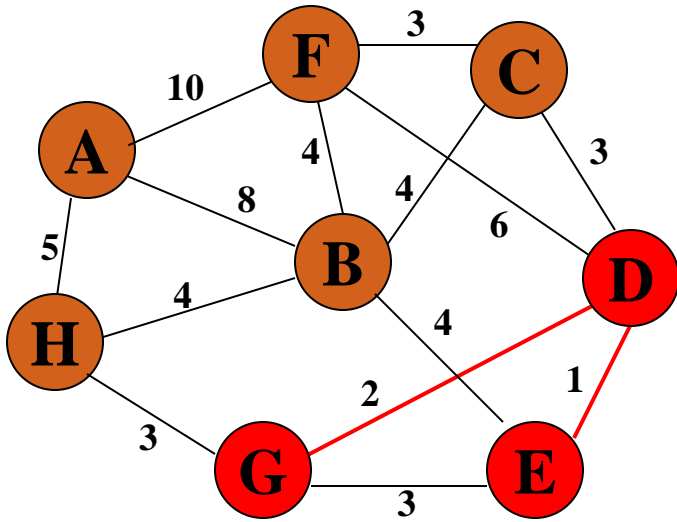


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<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	



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



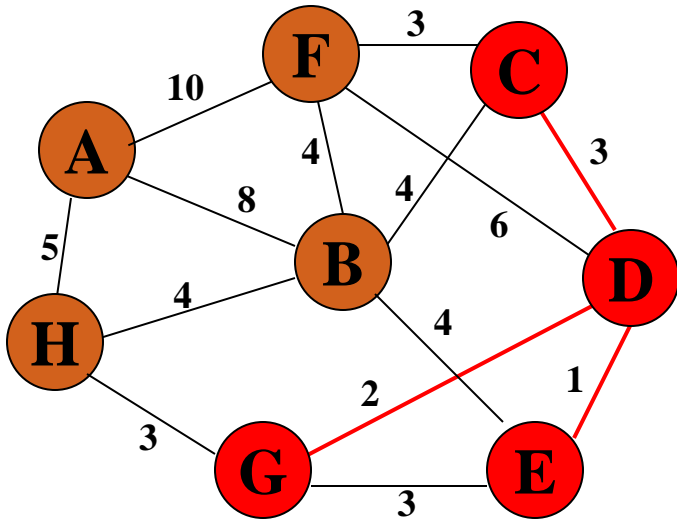
<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

<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



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

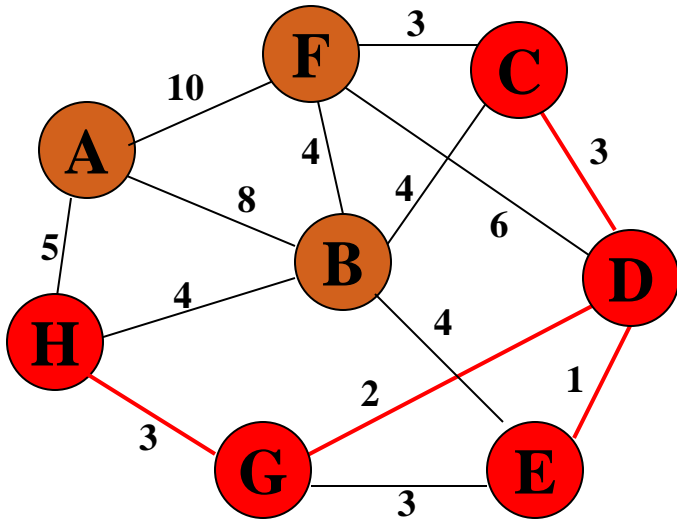


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	
(C,F)	3	
(B,C)	4	

<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	



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

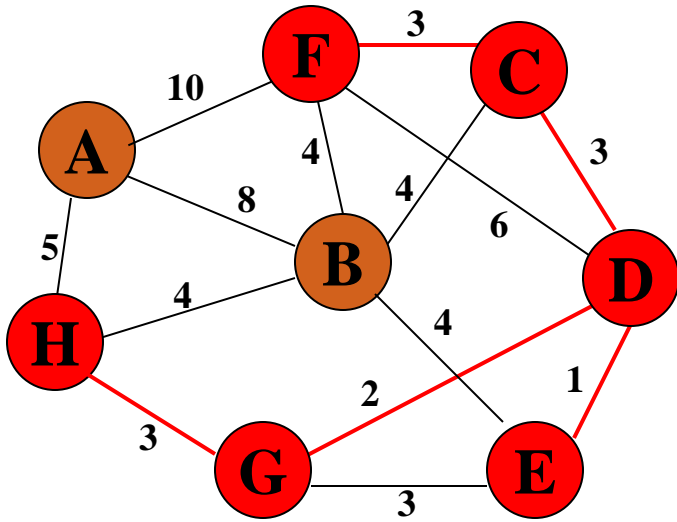


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	
(B,C)	4	

<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	



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

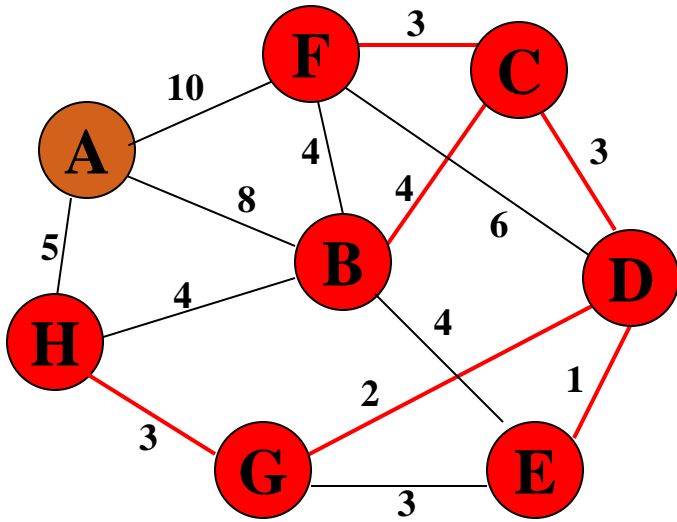


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	

<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	



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

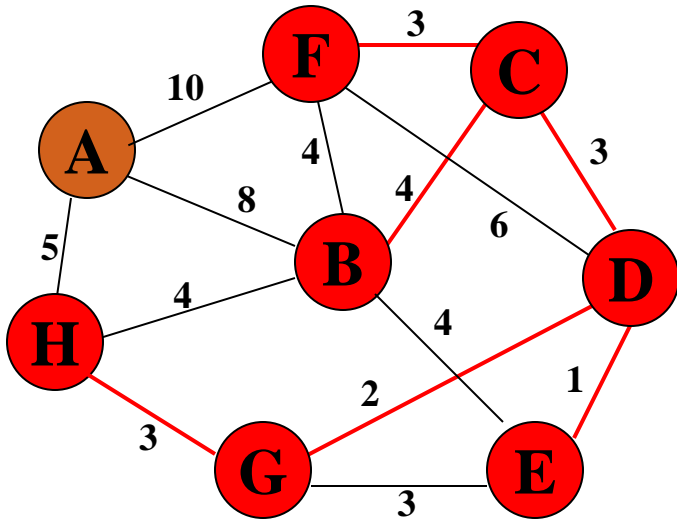


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<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	



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

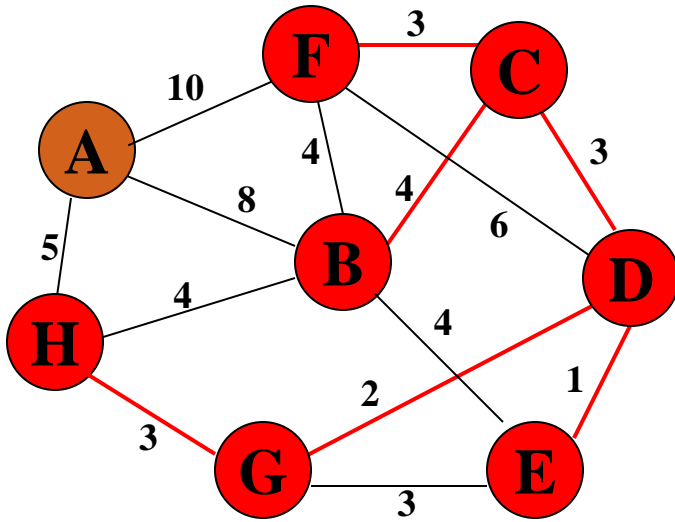


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<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	



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

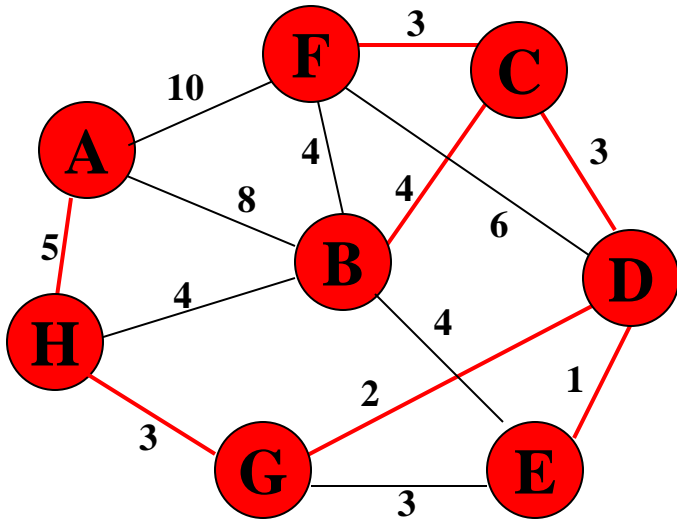


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<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	



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

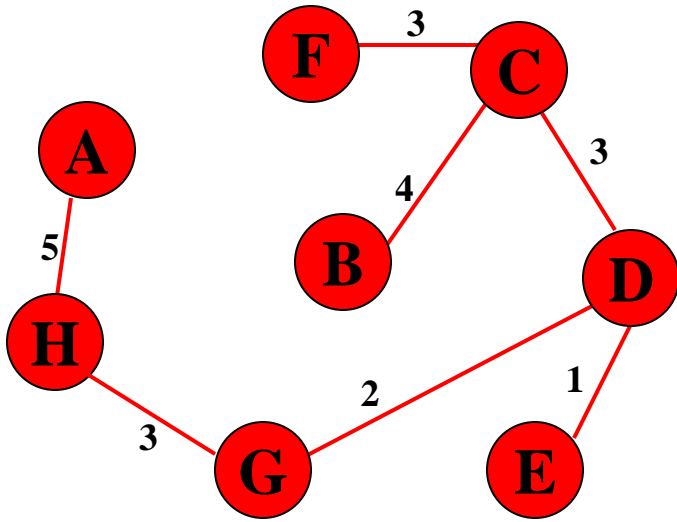


<i>edge</i>	d_v	
(D,E)	1	✓
(D,G)	2	✓
(E,G)	3	✗
(C,D)	3	✓
(G,H)	3	✓
(C,F)	3	✓
(B,C)	4	✓

<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	



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

$$\text{Total Cost} = \sum d_v = 21$$



DETECTING CYCLES

- Use Disjoint Sets (Chapter 8)

