Algorithm Design Approaches1

ESC108A Elements of Computer Science and Engineering

B. Tech. 2017

Course Leaders:

Roopa G.

Ami Rai E.

Chaitra S.

Objectives

- At the end of this lecture, student will be able to
  – list the important algorithmic approaches
  – Understand and apply Dijikstra’s Algorithm
  – Understand and apply Prim’s Algorithm
  – Understand and apply Kruskal’s Algorithm

Contents

- Dijikstra’s Algorithm
- Prims Algorithm
- Kruskal’s Algorithm

Dijkstra's algorithm

- **Dijkstra's algorithm** - a solution to the single-source shortest path problem in graph theory

- Works on both directed and undirected graphs
- However, all edges must have nonnegative weights

- Approach: Greedy

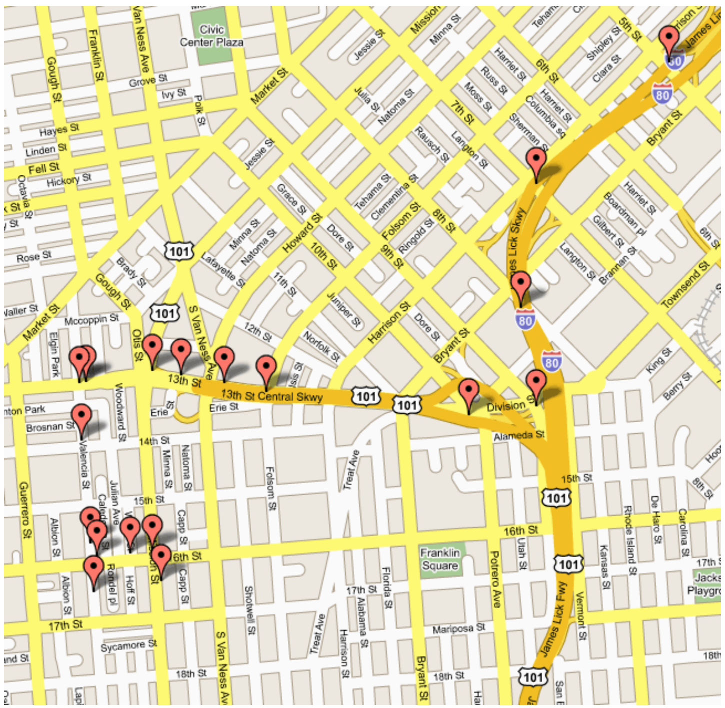
- Input: Weighted graph G={E,V} and source vertex $v \in V$, such that all edge weights are nonnegative
-
- Output: Lengths of shortest paths (or the shortest paths themselves) from a given source vertex $v \in V$ to all other vertices

DIJKSTRA'S ALGORITHM - WHY USE IT?

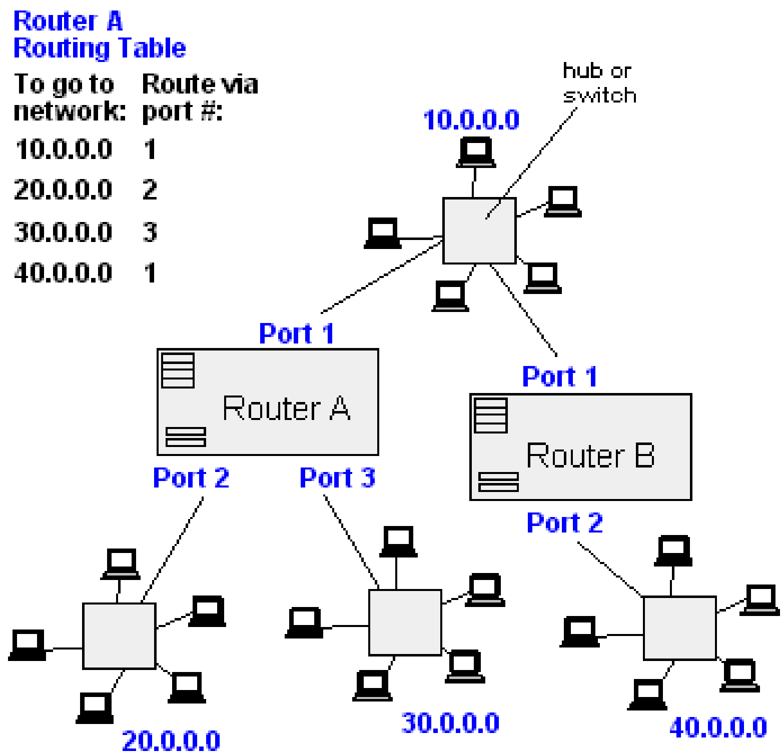
- As mentioned, Dijkstra's algorithm calculates the shortest path to every vertex from a given vertex
- However, it is as computationally expensive to calculate the shortest path from vertex *u* to every vertex using Dijkstra's as it is to calculate the shortest path to some particular vertex *v*
- Therefore, anytime we want to know the optimal path to some other vertex from a determined origin, we can use Dijkstra's algorithm

Applications of Dijkstra's Algorithm

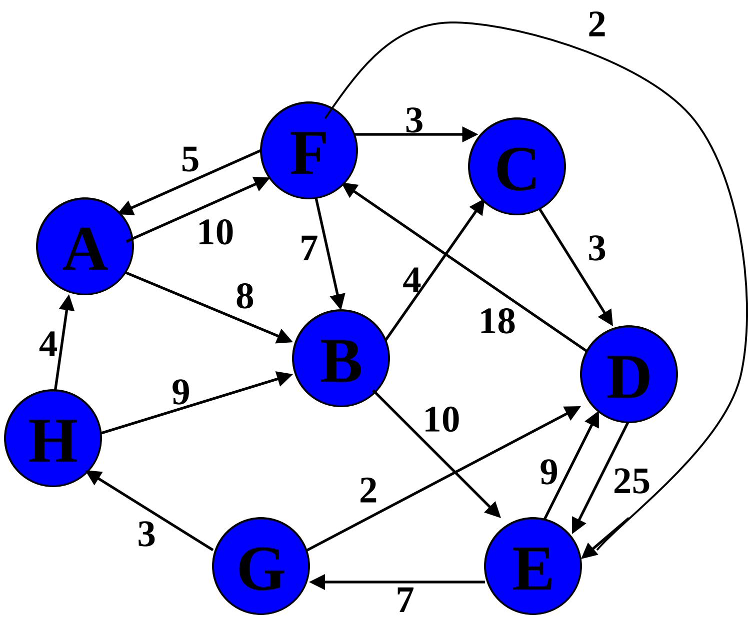
- Traffic Information Systems are most prominent use
- Mapping (Map Quest, Google Maps)
- Routing Systems



From Computer Desktop Encyclopedia



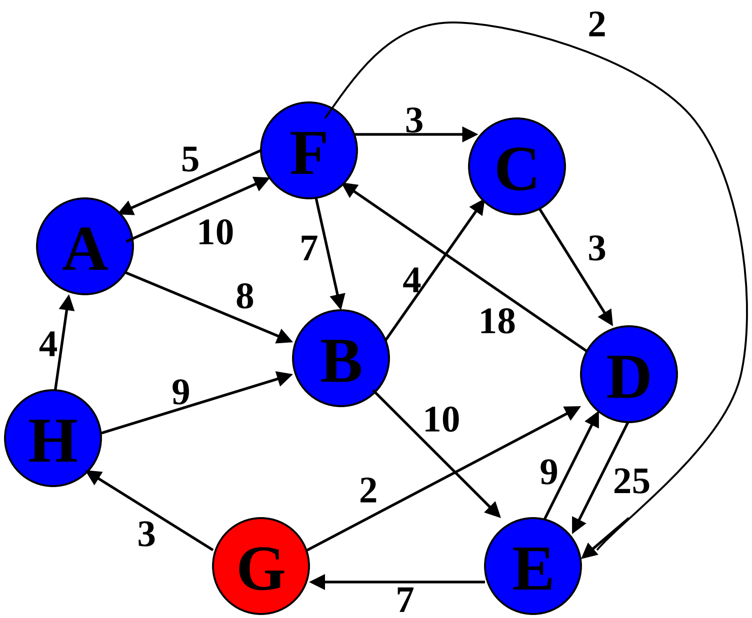
Example-Walk-Through



Initialize array

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

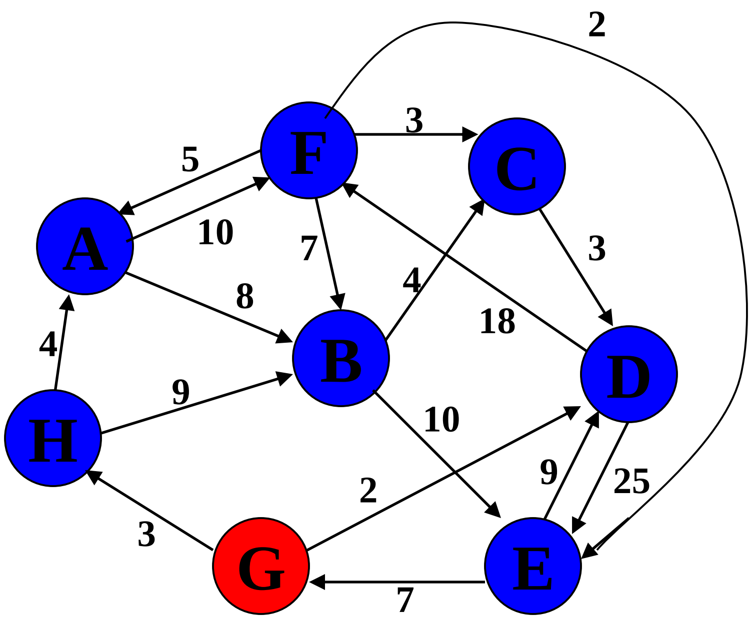
Walk-Through contd.



Start with G

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

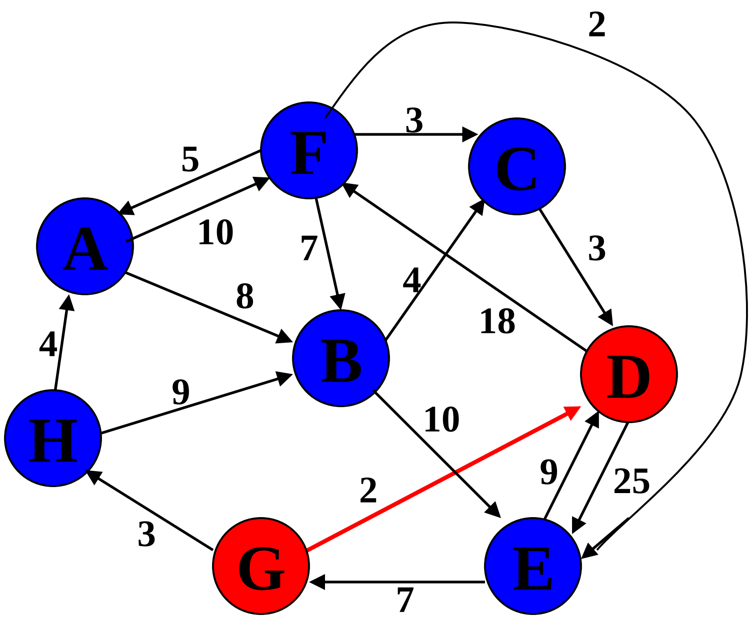
Walk-Through contd.



Update unselected nodes

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

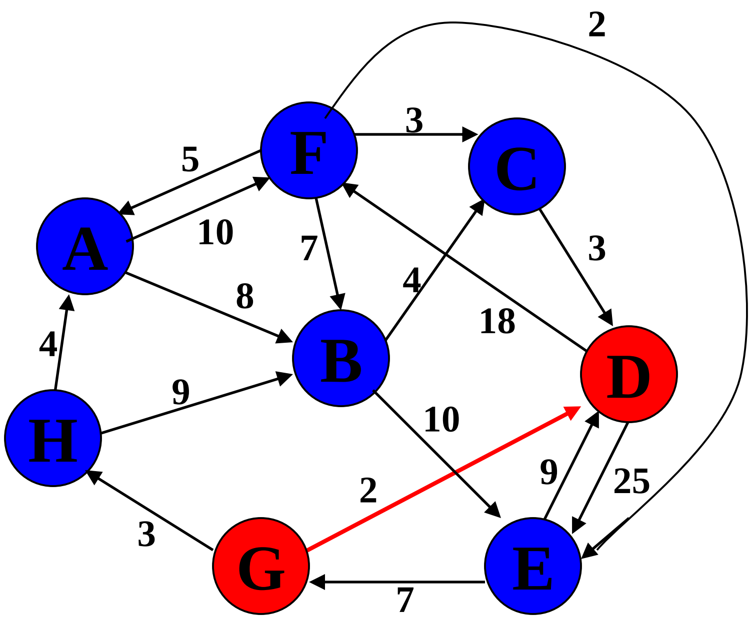
Walk-Through contd.



Select minimum distance

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

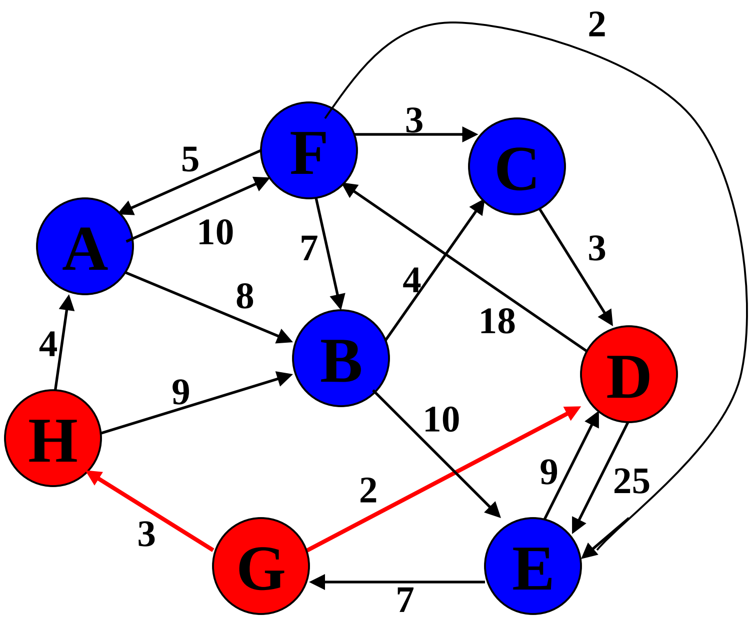
Walk-Through contd.



Update unselected nodes

	K	d_v	p_v
A			
B			
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H		3	G

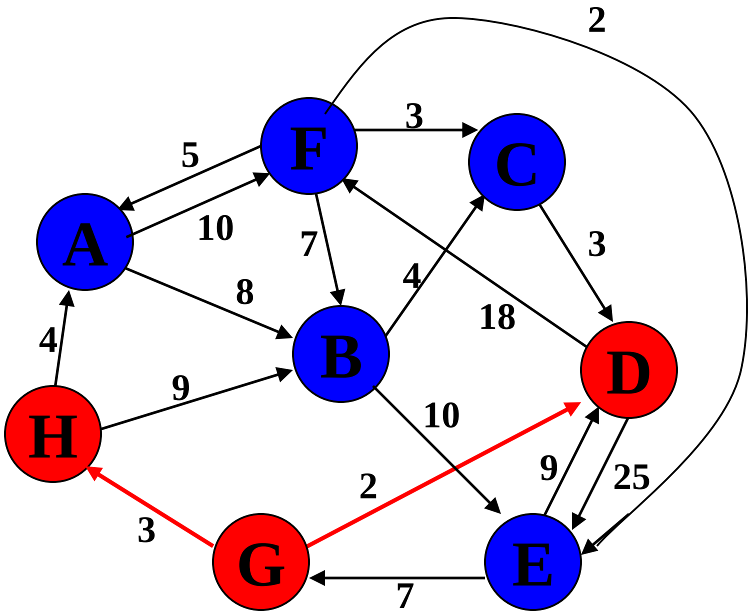
Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A			
B			
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

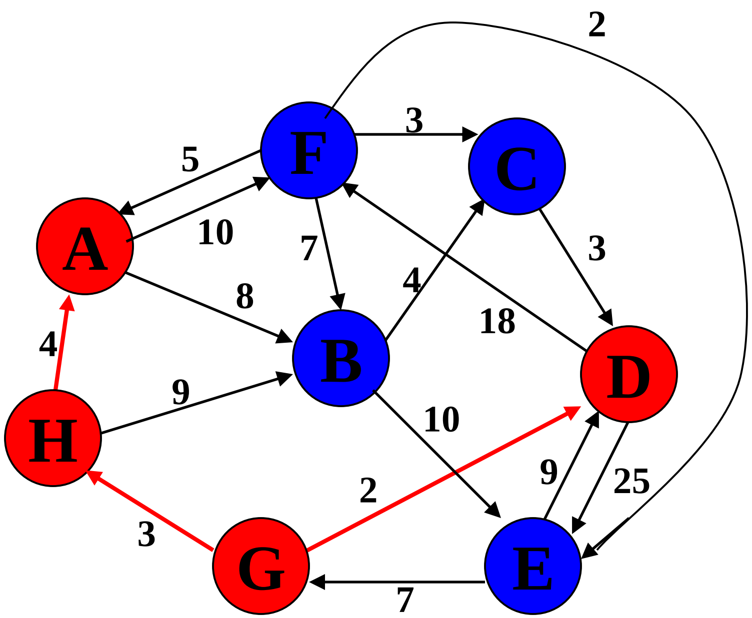
Walk-Through contd.



Update unselected nodes

	K	d_v	p_v
A		7	H
B		12	H
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

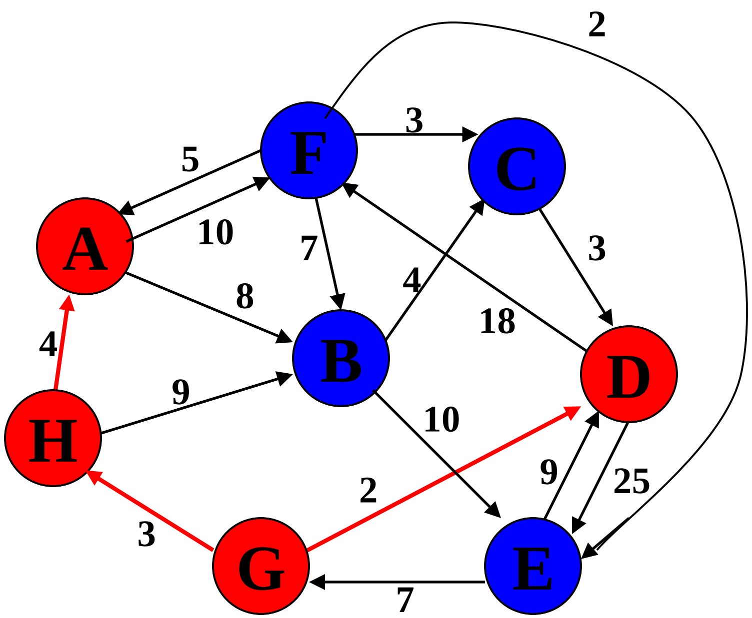
Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A	T	7	H
B		12	H
C			
D	T	2	G
E		27	D
F		20	D
G	T	0	–
H	T	3	G

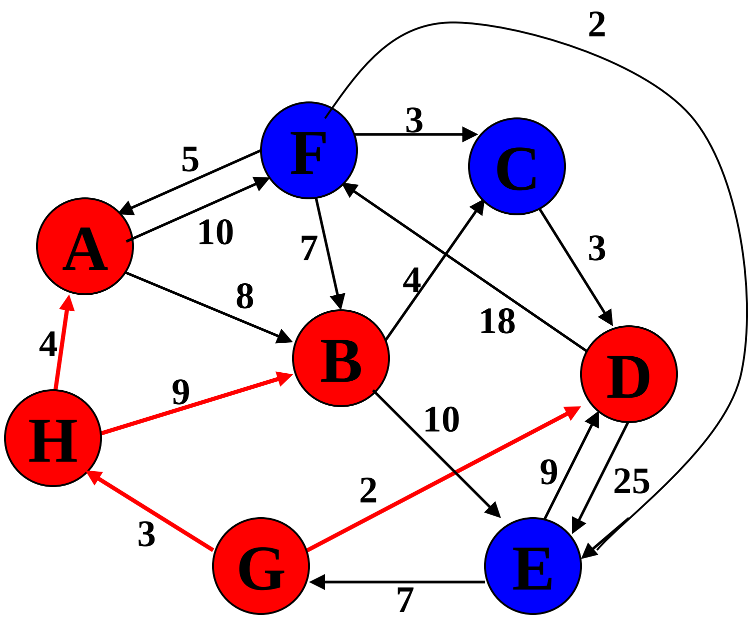
Walk-Through contd.



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B		12	H
C			
D	T	2	G
E		27	D
F		17	A
G	T	0	–
H	T	3	G

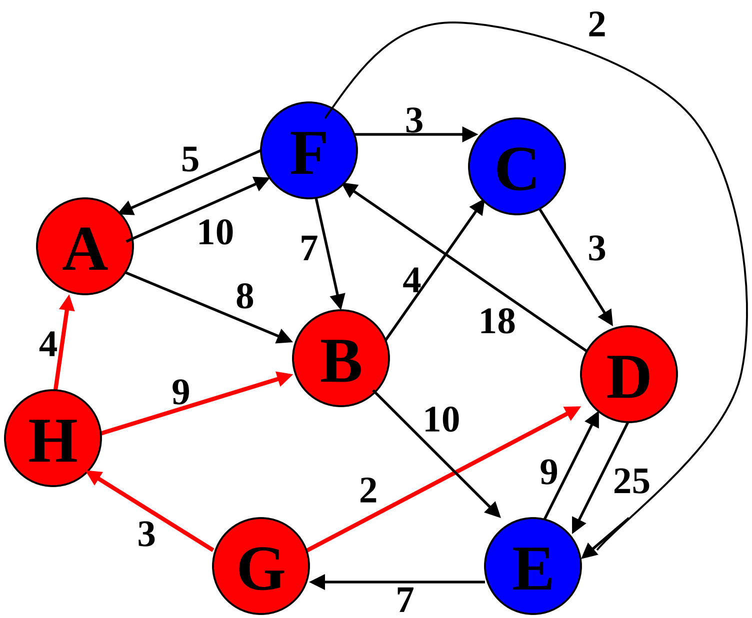
Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C			
D	T	2	G
E		27	D
F		17	A
G	T	0	–
H	T	3	G

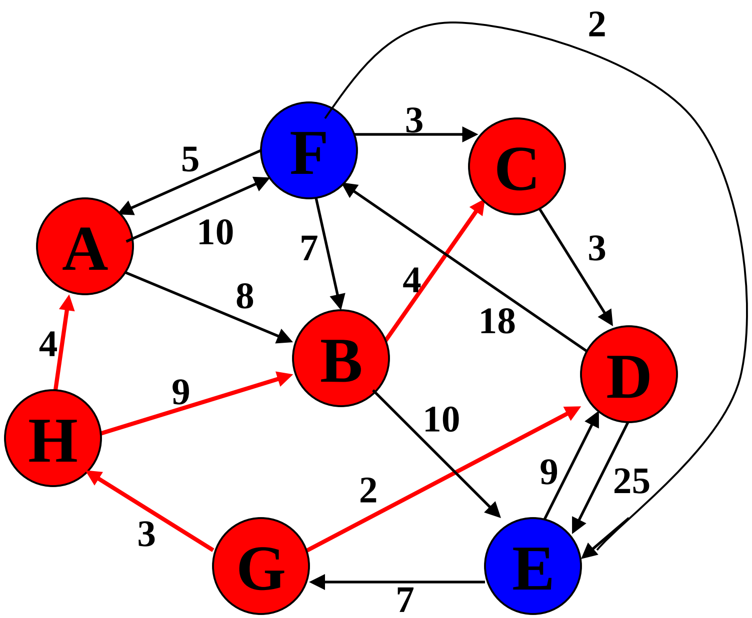
Walk-Through contd.



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C		16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

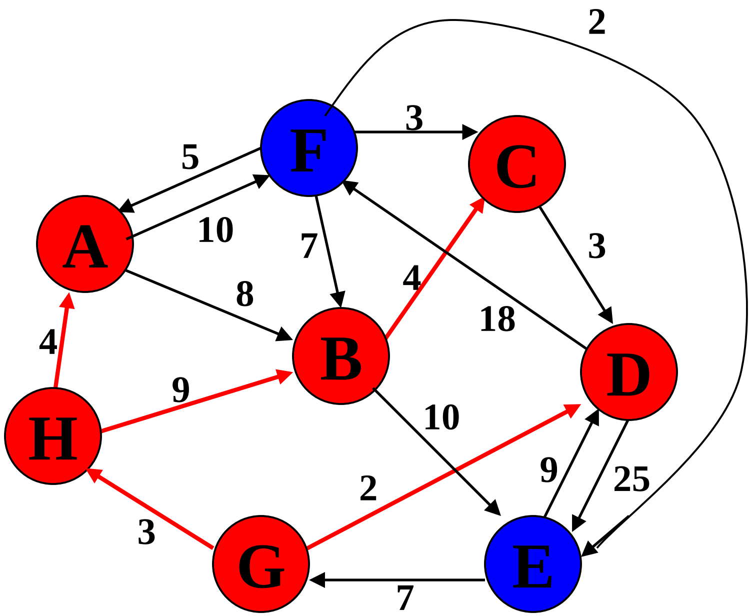
Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

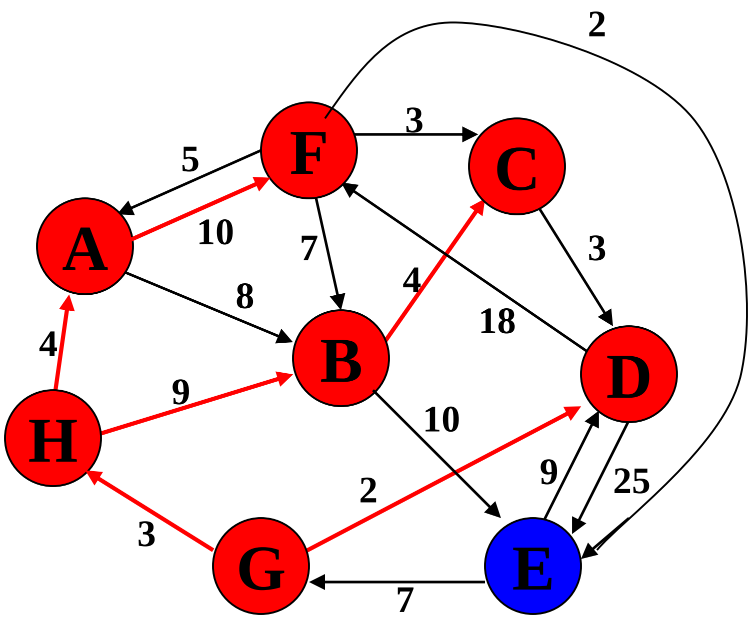
Walk-Through contd.



- Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F		17	A
G	T	0	–
H	T	3	G

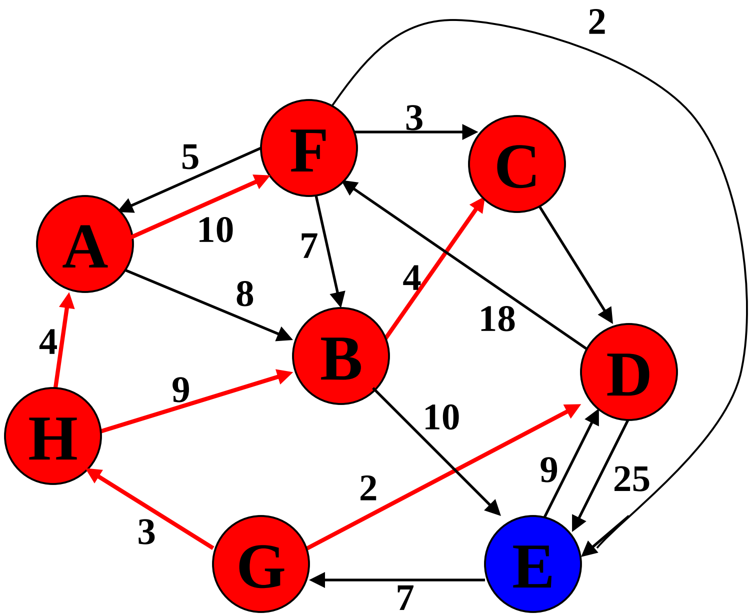
Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		22	B
F	T	17	A
G	T	0	–
H	T	3	G

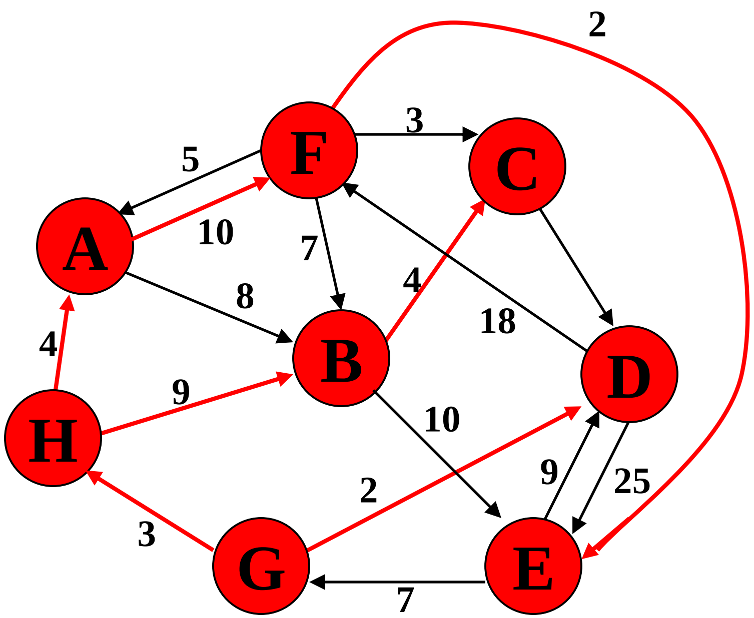
Walk-Through contd.



Update unselected nodes

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E		19	F
F	T	17	A
G	T	0	–
H	T	3	G

Walk-Through contd.



Select minimum distance

	K	d_v	p_v
A	T	7	H
B	T	12	H
C	T	16	B
D	T	2	G
E	T	19	F
F	T	17	A
G	T	0	–
H	T	3	G

Done

Minimum Spanning Tree

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

Prim’s and Kruskal’s Algorithms

- 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

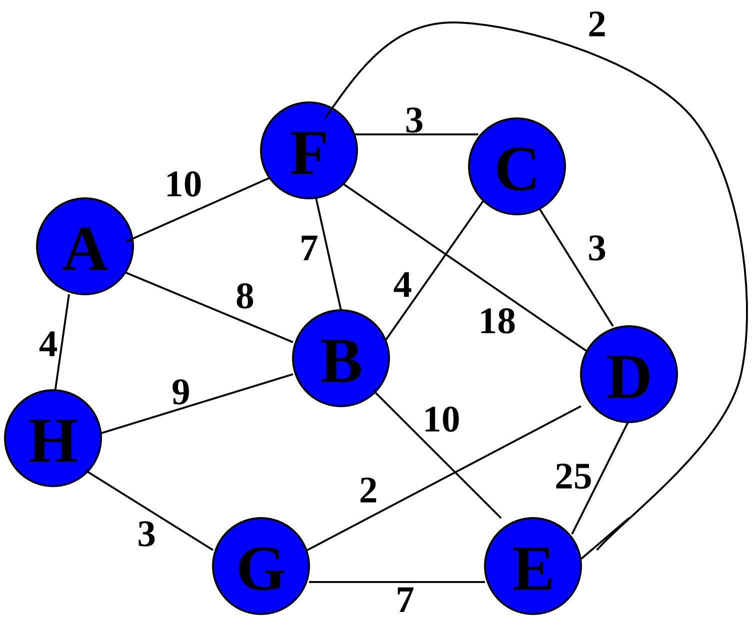
Some Applications

- Taxonomy
- Clustering Analysis
- Traveling Salesman Problem Approximation
- In the design of electronic circuitry

Prim’s Algorithm

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

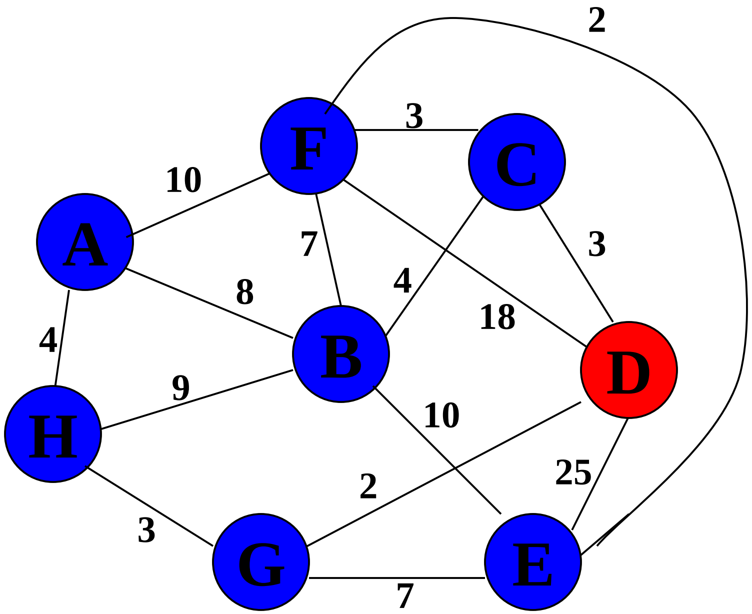
Walk-Through



Initialize array

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

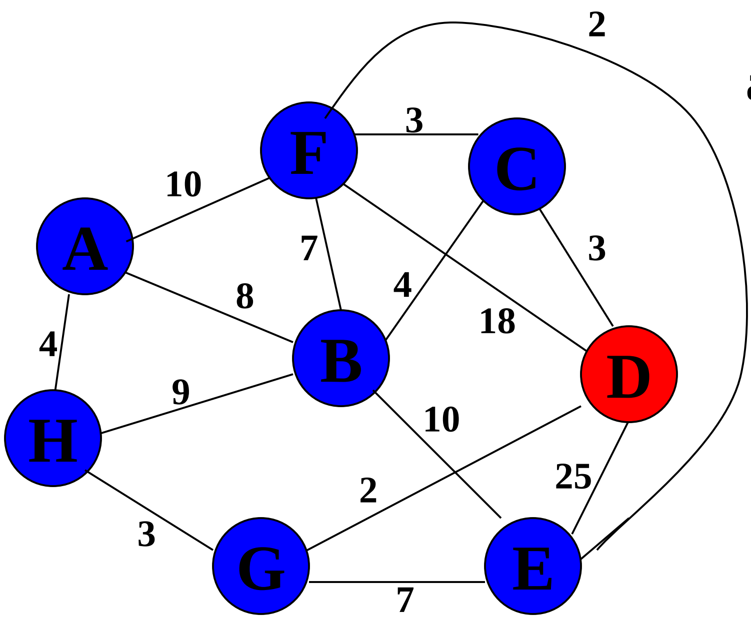
Walk-Through contd.



Start with any node, say D

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

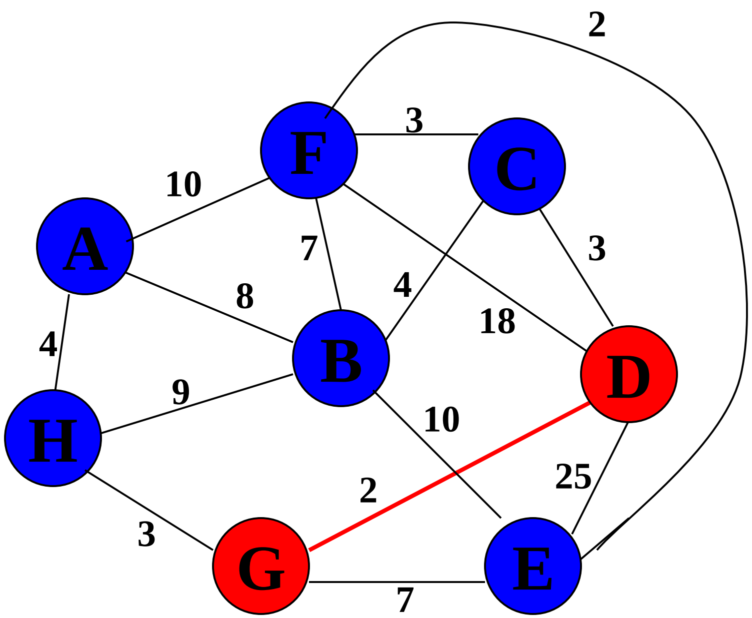
Walk-Through contd.



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			

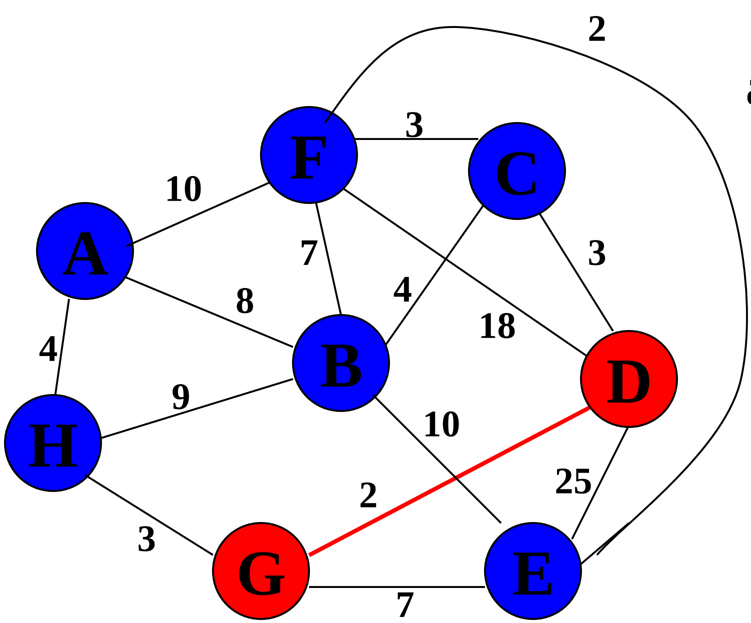
Walk-Through contd.



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			

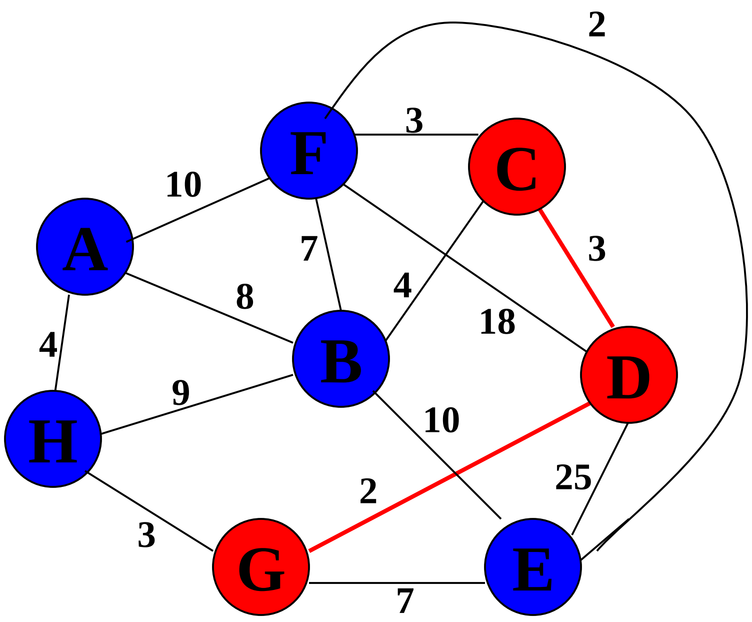
Walk-Through contd.



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

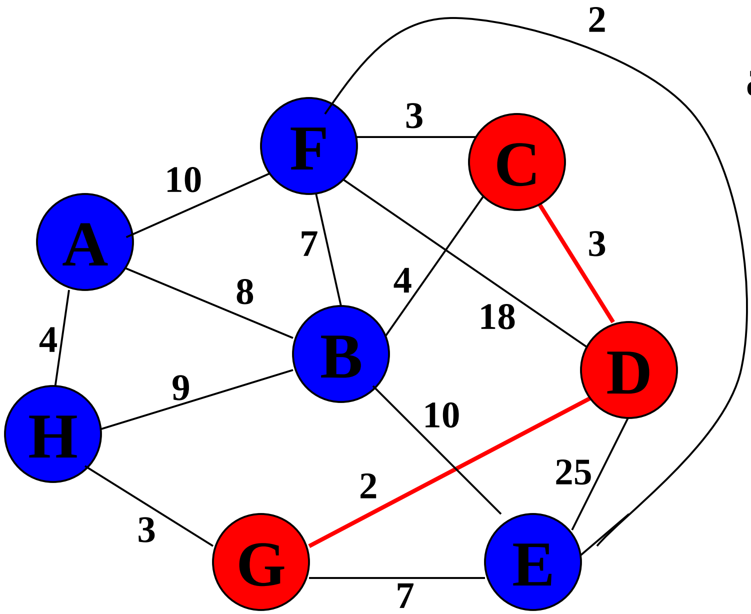
Walk-Through contd.



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

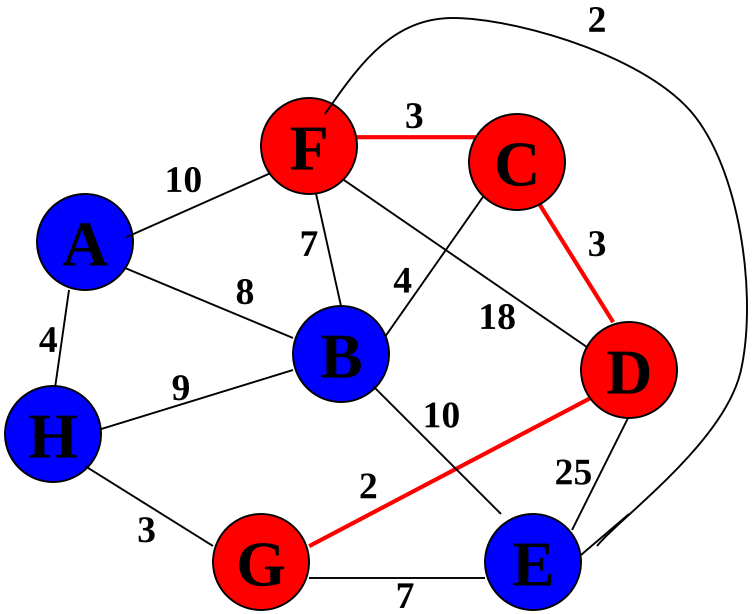
Walk-Through contd.



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

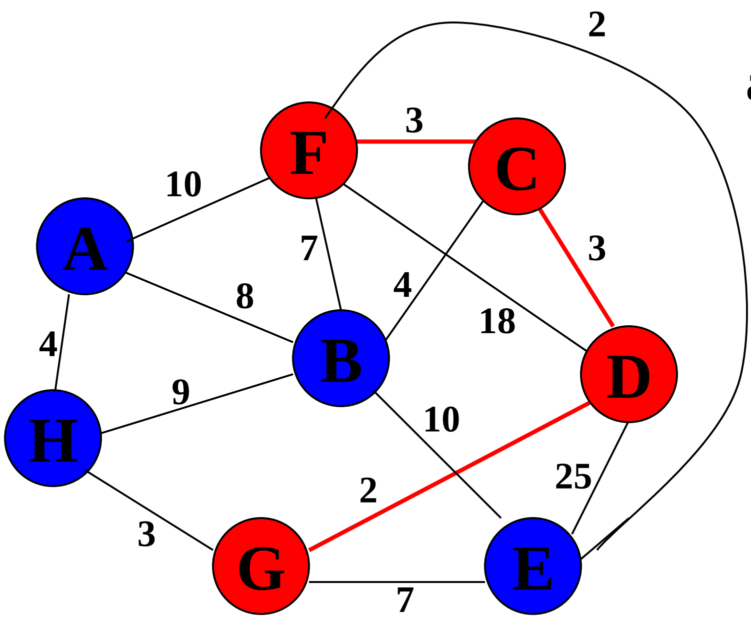
Walk-Through contd.



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

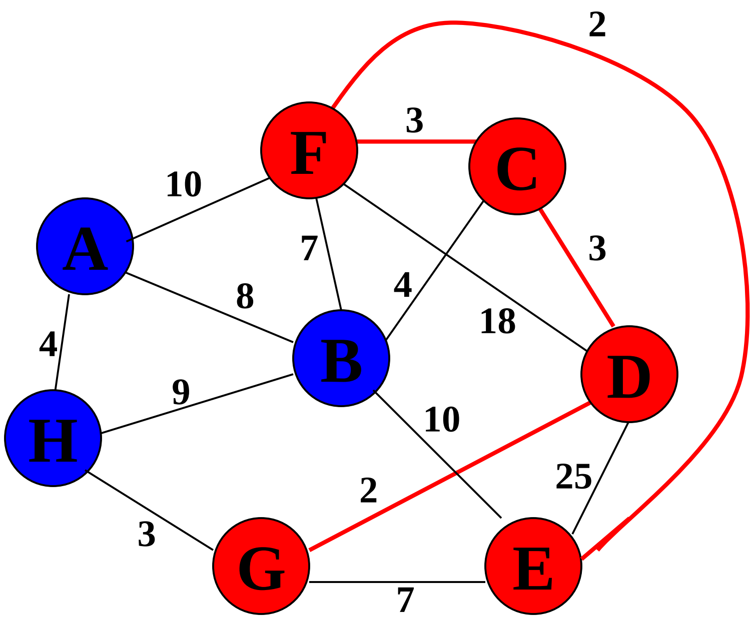
Walk-Through contd.



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

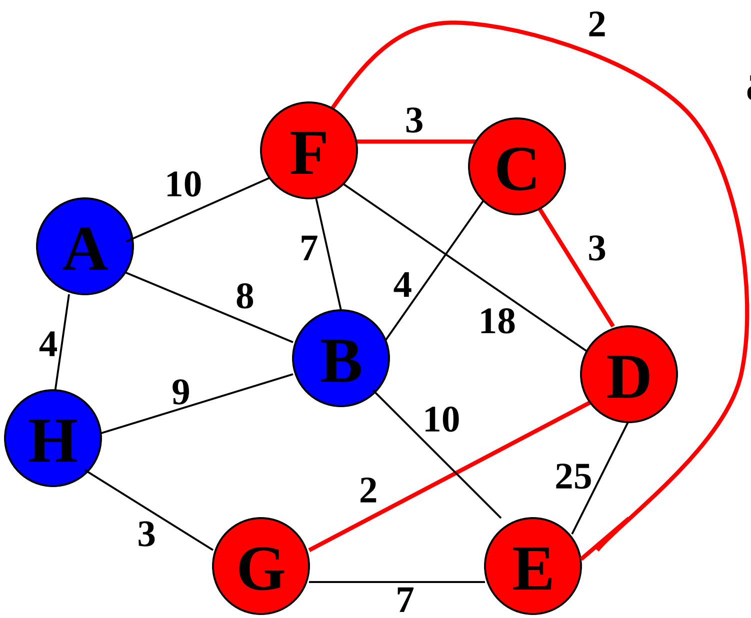
Walk-Through contd.



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

Walk-Through contd.

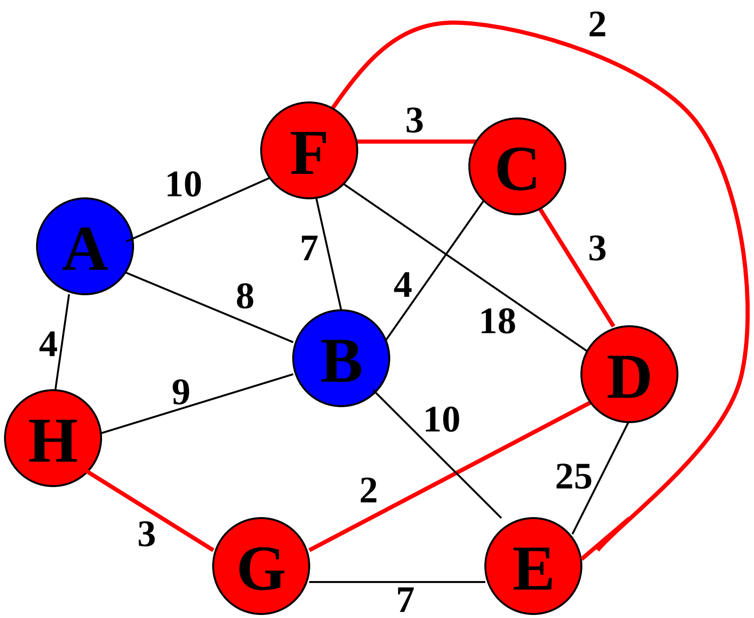


Update distances of adjacent, unselected nodes

	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

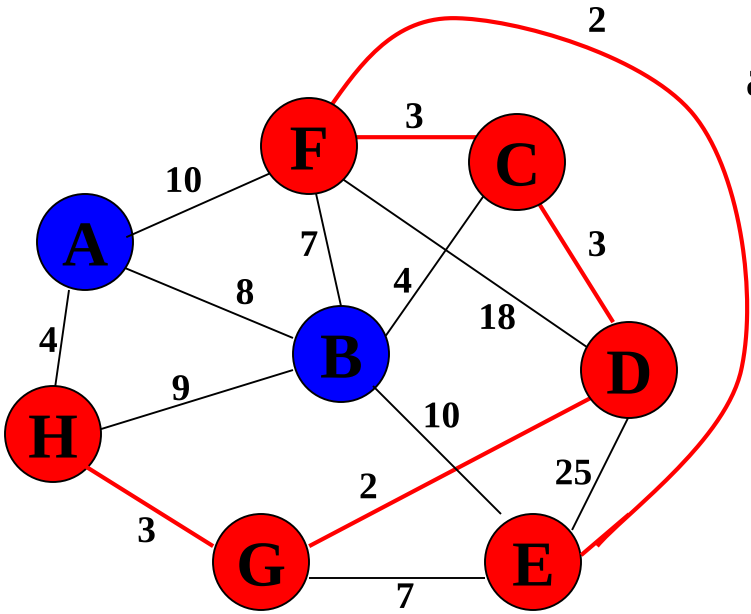
Walk-Through contd.



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

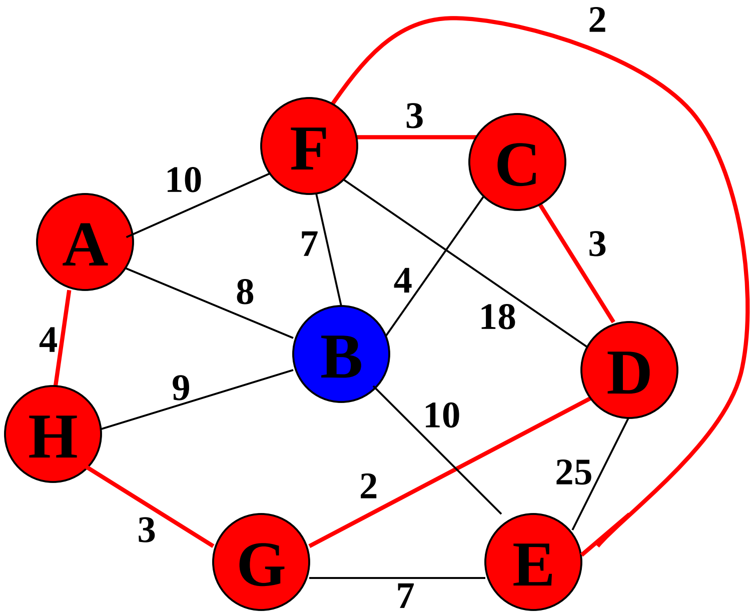
Walk-Through contd.



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

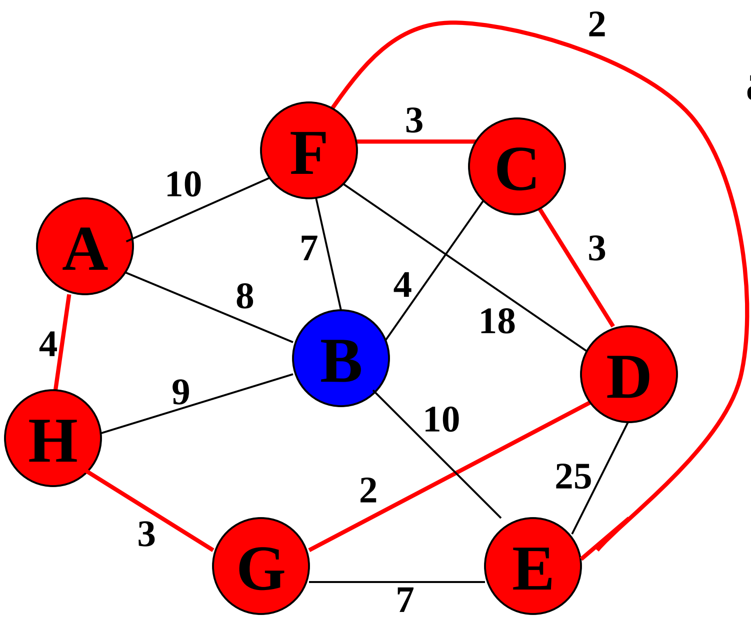
Walk-Through contd.



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

Walk-Through contd.

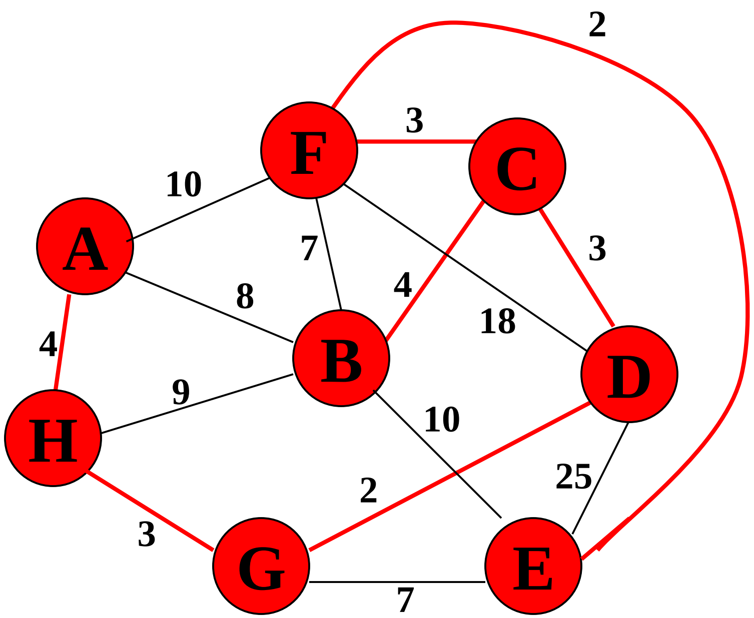


Update distances of adjacent, unselected nodes

	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

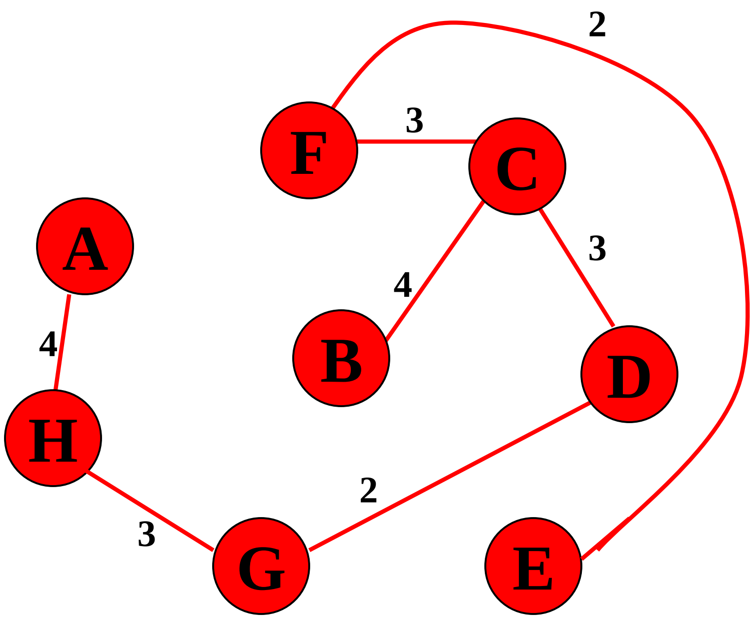
Walk-Through contd.



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

Walk-Through contd.



Cost of Minimum Spanning Tree = $\Sigma\, d_v$ = **21**

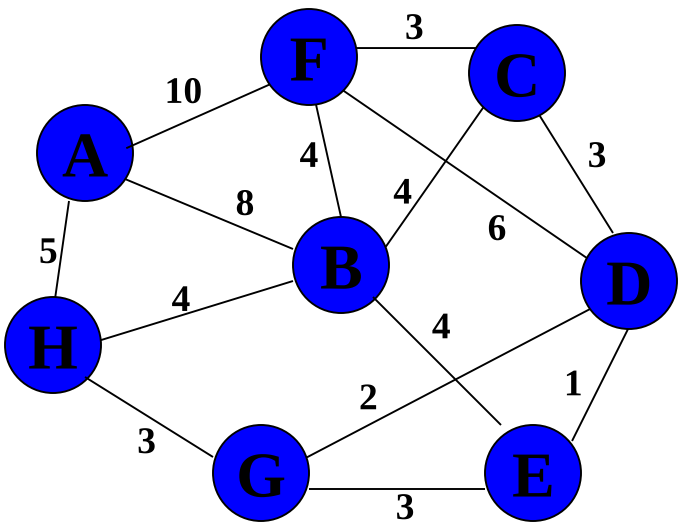
	K	d_v	p_v
A	T	4	H
B	T	4	C
C	T	3	D
D	T	0	–
E	T	2	F
F	T	3	C
G	T	2	D
H	T	3	G

Done

Kruskal's Algorithm

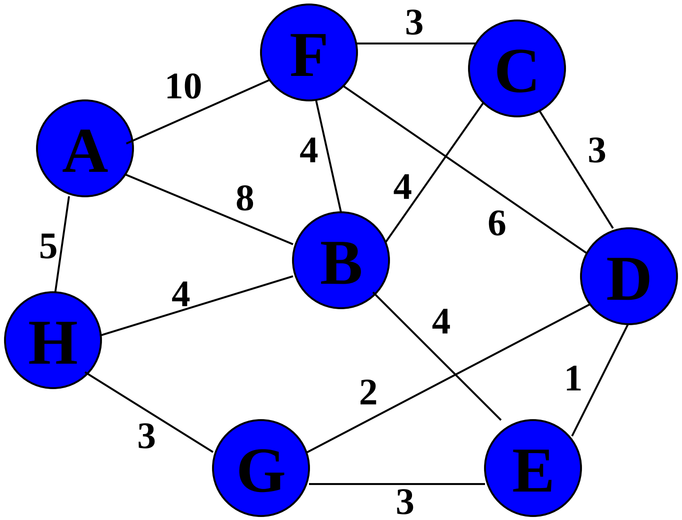
- Work with edges, rather than nodes
- Two steps:
  - Sort edges by increasing edge weight
  - Select the first |V| – 1 edges that do not generate a cycle

Walk-Through



Consider an undirected, weight graph

Walk-Through contd.

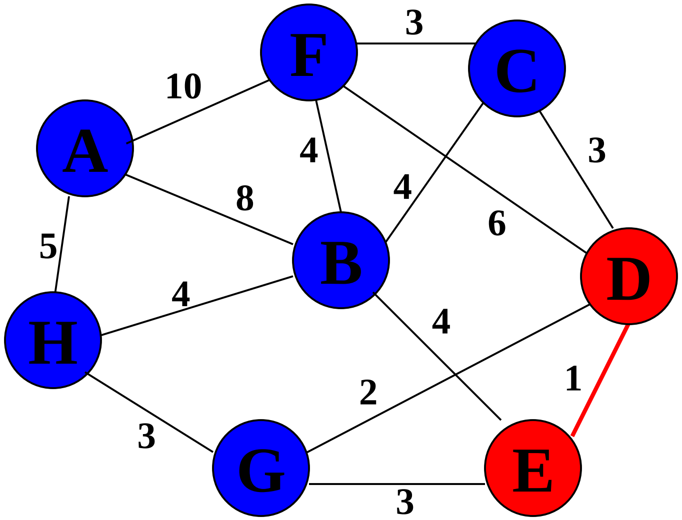


Sort the edges by increasing edge weight

edge	d_v	
(D,E)	1	
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

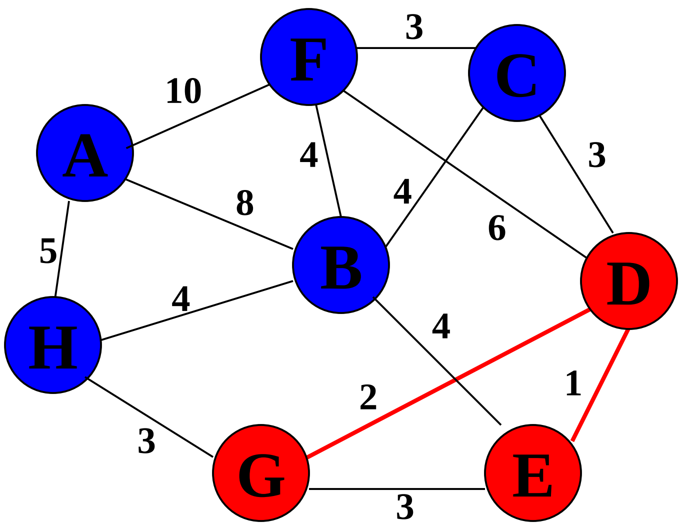


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

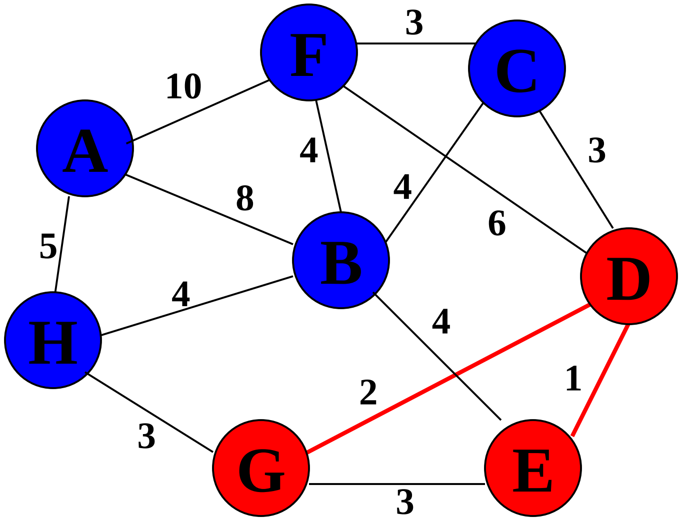


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.



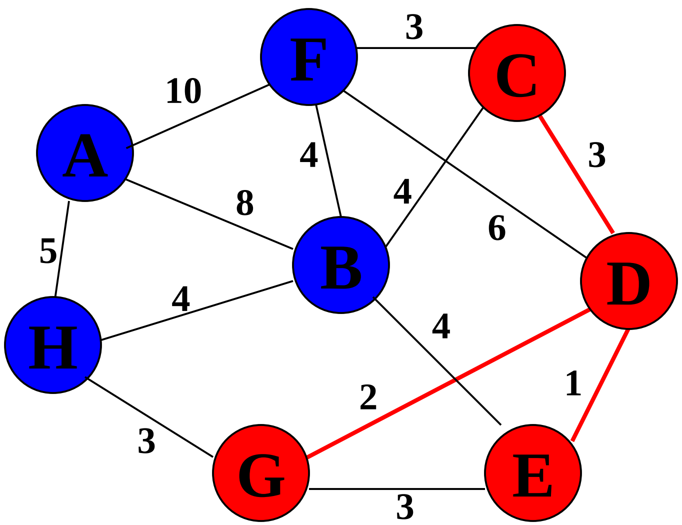
Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	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

Walk-Through contd.



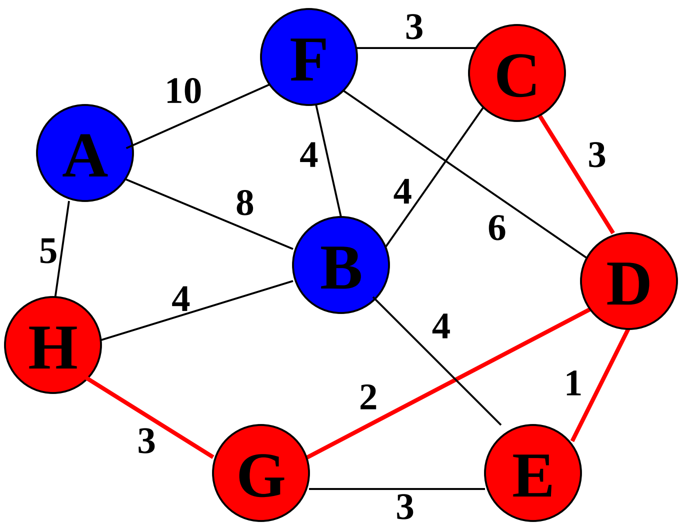
Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

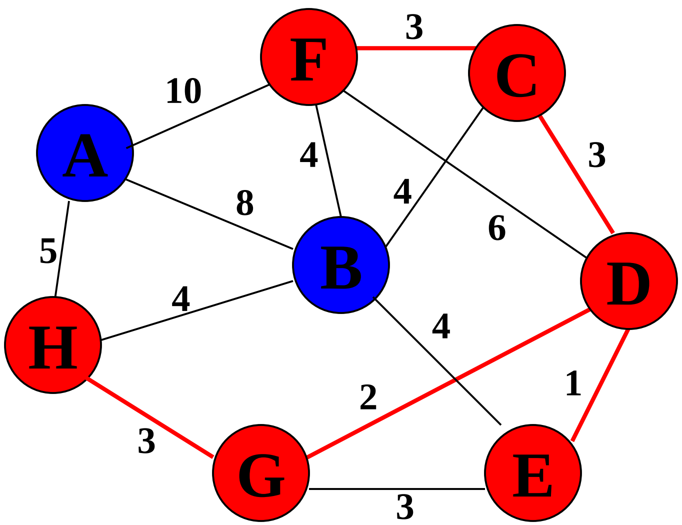
Select first |V|–1 edges which do not generate a cycle



edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

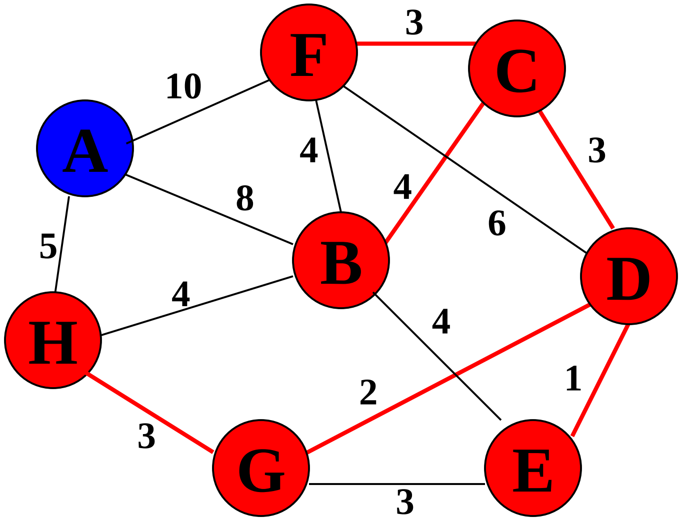


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

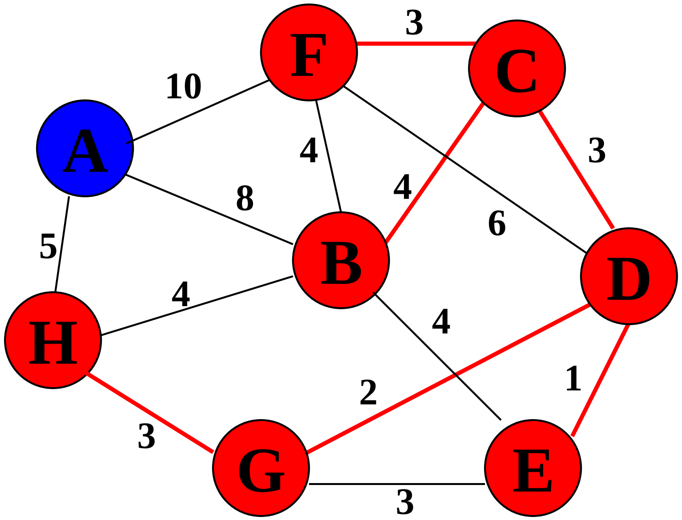


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.



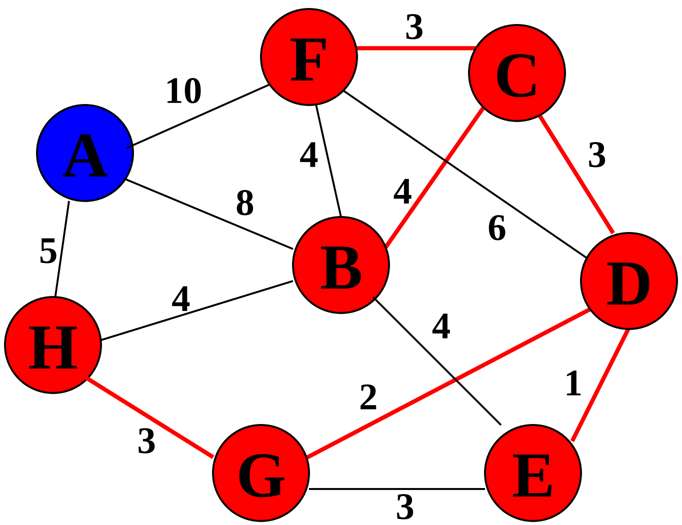
Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

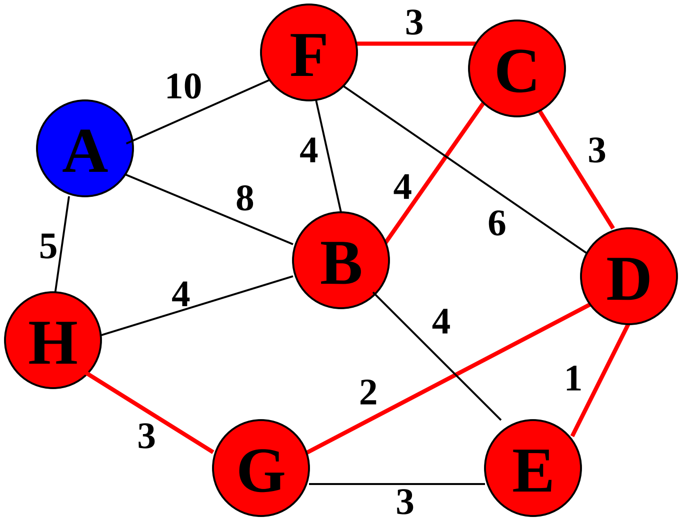
Select first |V|–1 edges which do not generate a cycle



edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

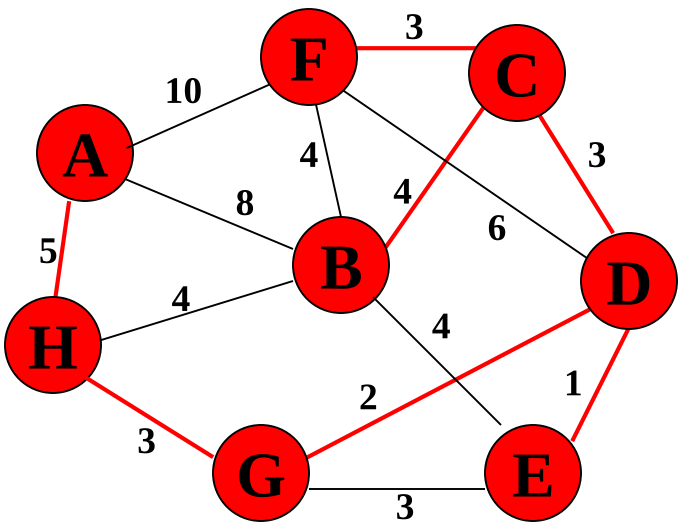


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.

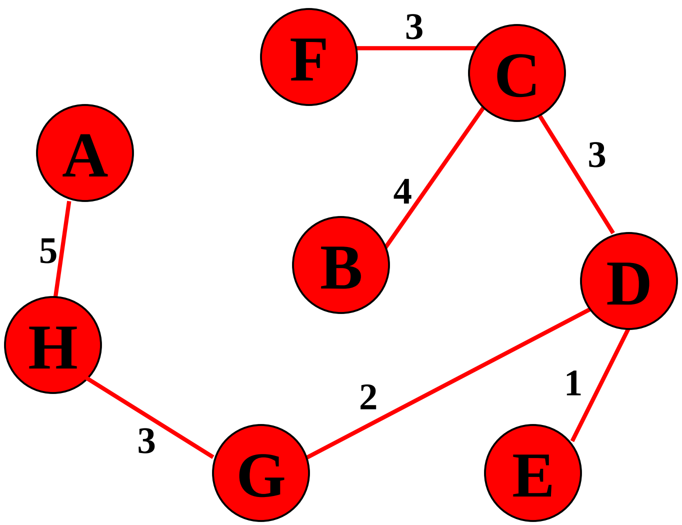


Select first |V|–1 edges which do not generate a cycle

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	
(A,B)	8	
(A,F)	10	

Walk-Through contd.



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

edge	d_v	
(D,E)	1	√
(D,G)	2	√
(E,G)	3	χ
(C,D)	3	√
(G,H)	3	√
(C,F)	3	√
(B,C)	4	√

edge	d_v	
(B,E)	4	χ
(B,F)	4	χ
(B,H)	4	χ
(A,H)	5	√
(D,F)	6	
(A,B)	8	
(A,F)	10	

$\left.\begin{matrix}(D,F)\\(A,B)\\(A,F)\end{matrix}\right\}$ not considered

Done

Total Cost = $\boldsymbol{\Sigma\, d_v = 21}$

Summary

- Greedy approach is a powerful technique to solve problem of finding optimal solutions
- Greedy approaches lead to efficient algorithms
- The general greedy technique involves step wise selection of the best possible alternatives available
- Dijkstra's algorithm is a solution to the single-source shortest path problem in graph theory
- Both Prim's and Kruskal's Algorithms work with undirected graphs

References

Dromey, R. (1982) *How To Solve it By Computer.* Noida: Pearson Education Inc.

Goodrich, M. T., and Tomasia, R. (2001) *Algorithm Design: Foundations, Analysis, and Internet Examples.* Wiley

Levitin, A. (2003) *Introduction to the Design and Analysis of Algorithms.* Addison-Wesley