Data Structures:
Graphs: Traversal, Minimum Spanning Tree, Graph Traversal Algorithms

Won Kim

(Lecture by Youngmin Oh)

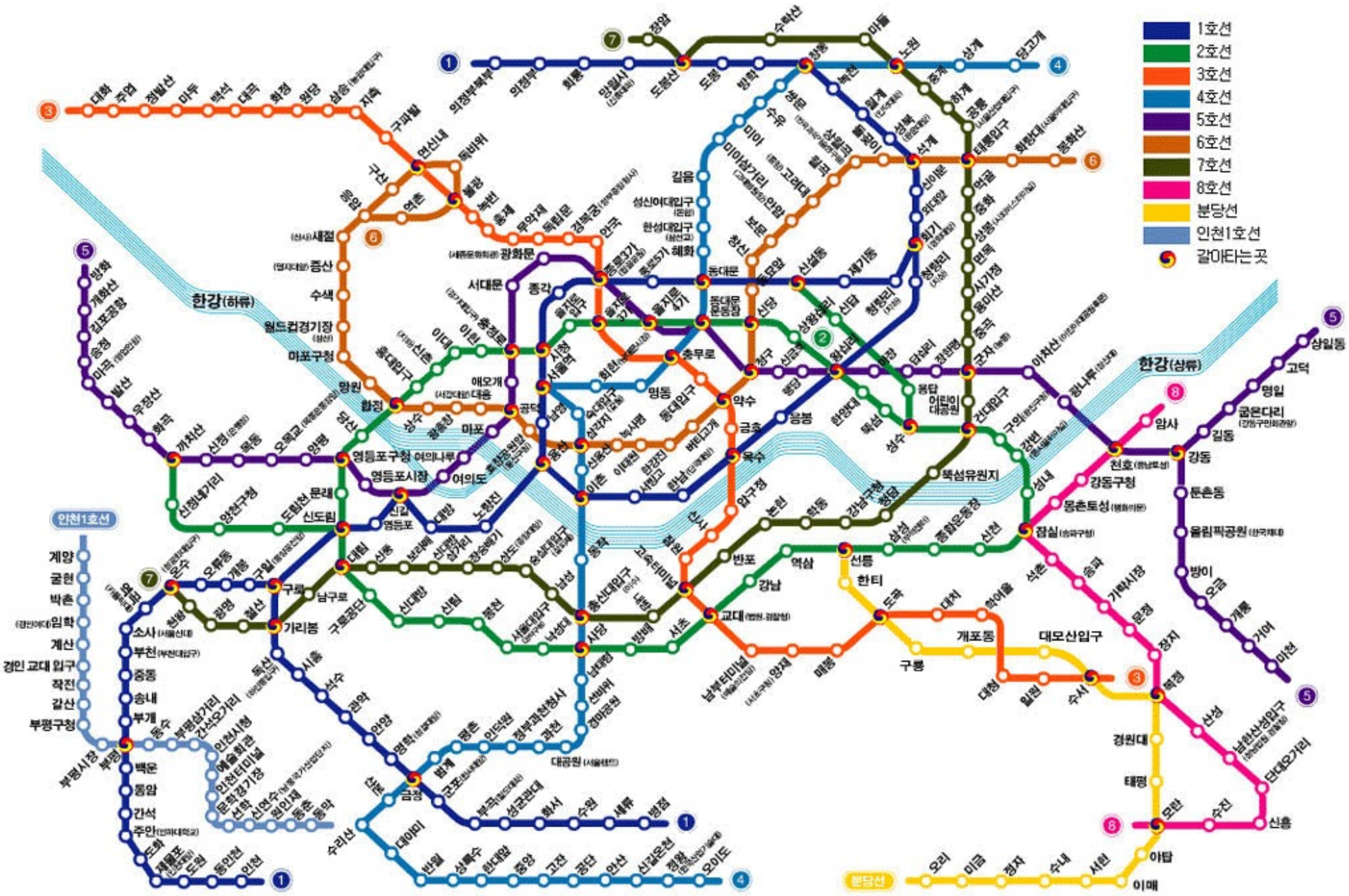
Spring 2022

Graphs

Graphs

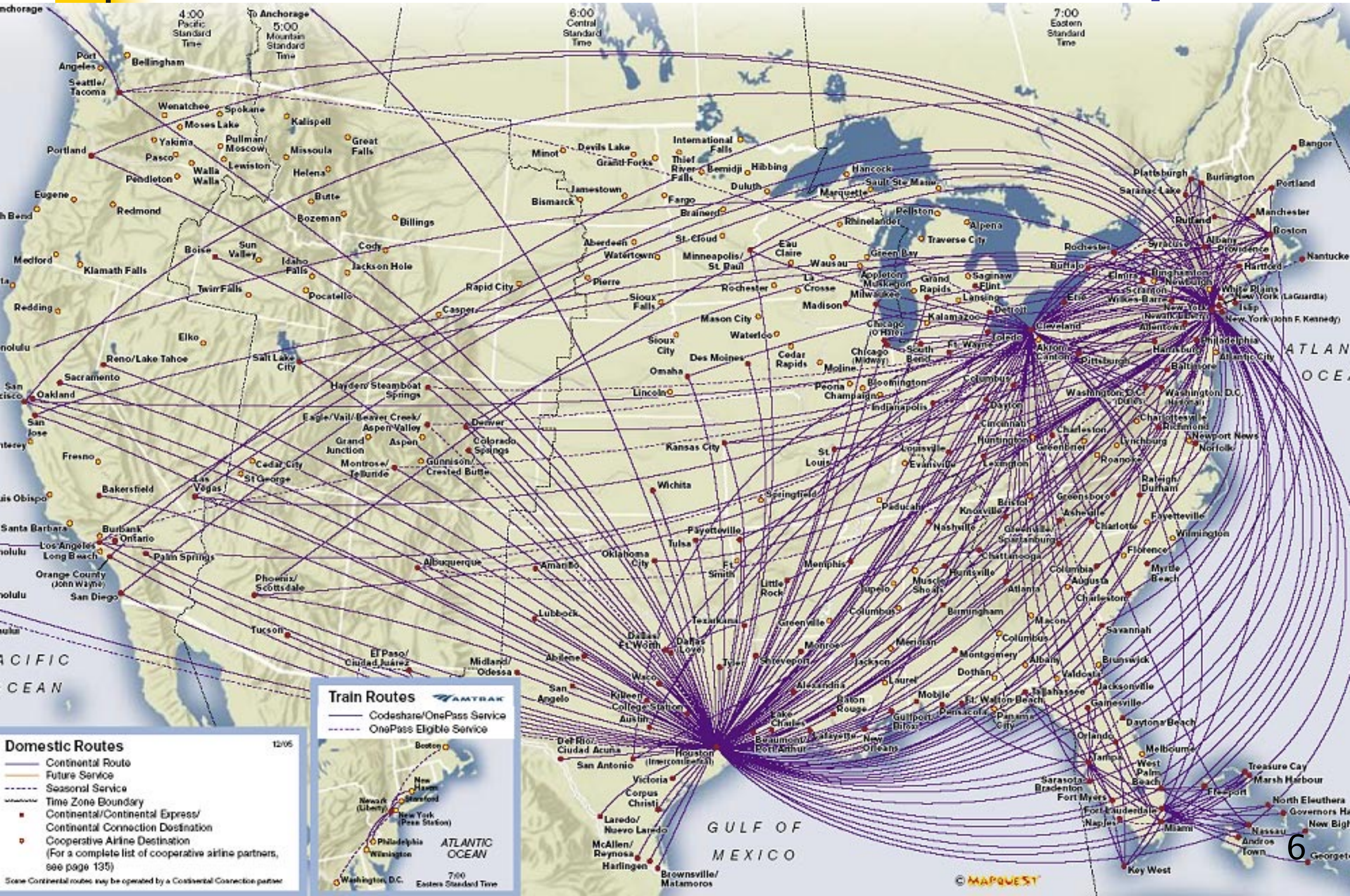
- A tree is a special case of a graph
- Graphs are useful in modeling various problems, and devising algorithms for solving the problems.
  - Determining the cheapest airfare route
  - Determining the fastest route for routing messages in a telecommunication system
  - Determining the cheapest route for transmitting oil through a nationwide network of oil pipelines (USA)
  - Determining the most efficient way to complete a multi-task project
  - ….

Real-World Graph Examples

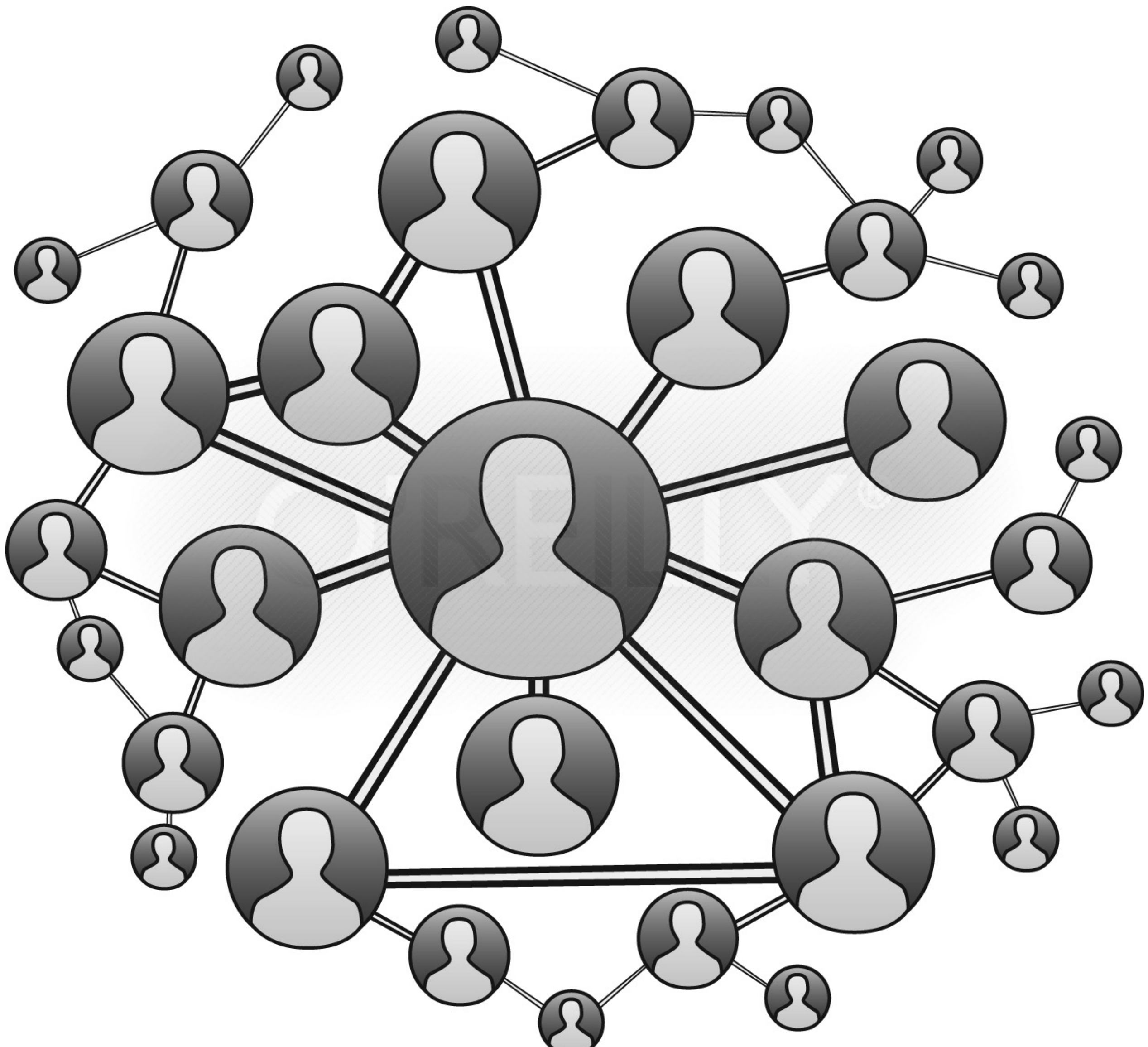


Asiana Airlines Korea Route Map
(no need for a graph algorithm ^ ^)

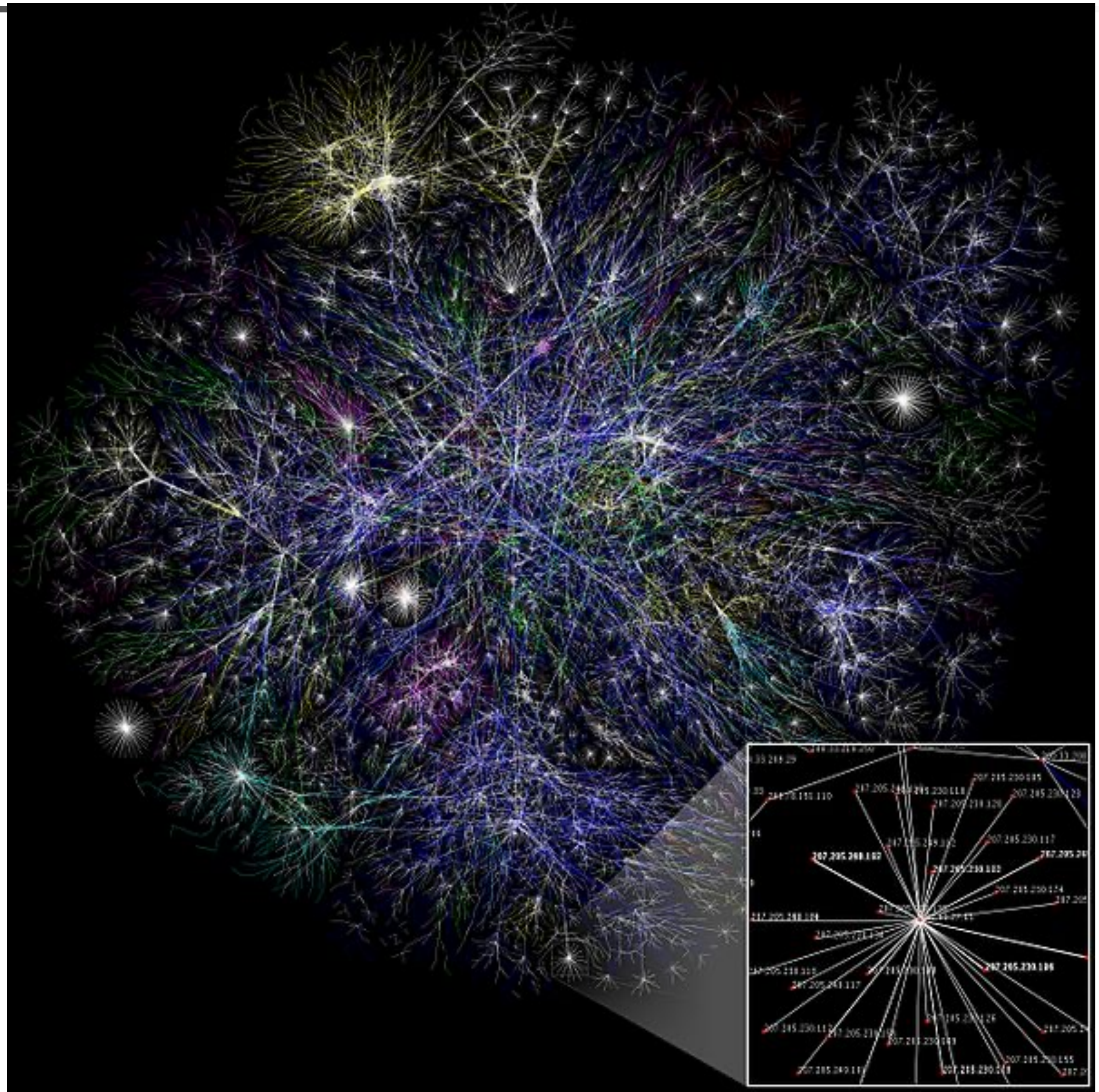
Continental Airlines US Route Map



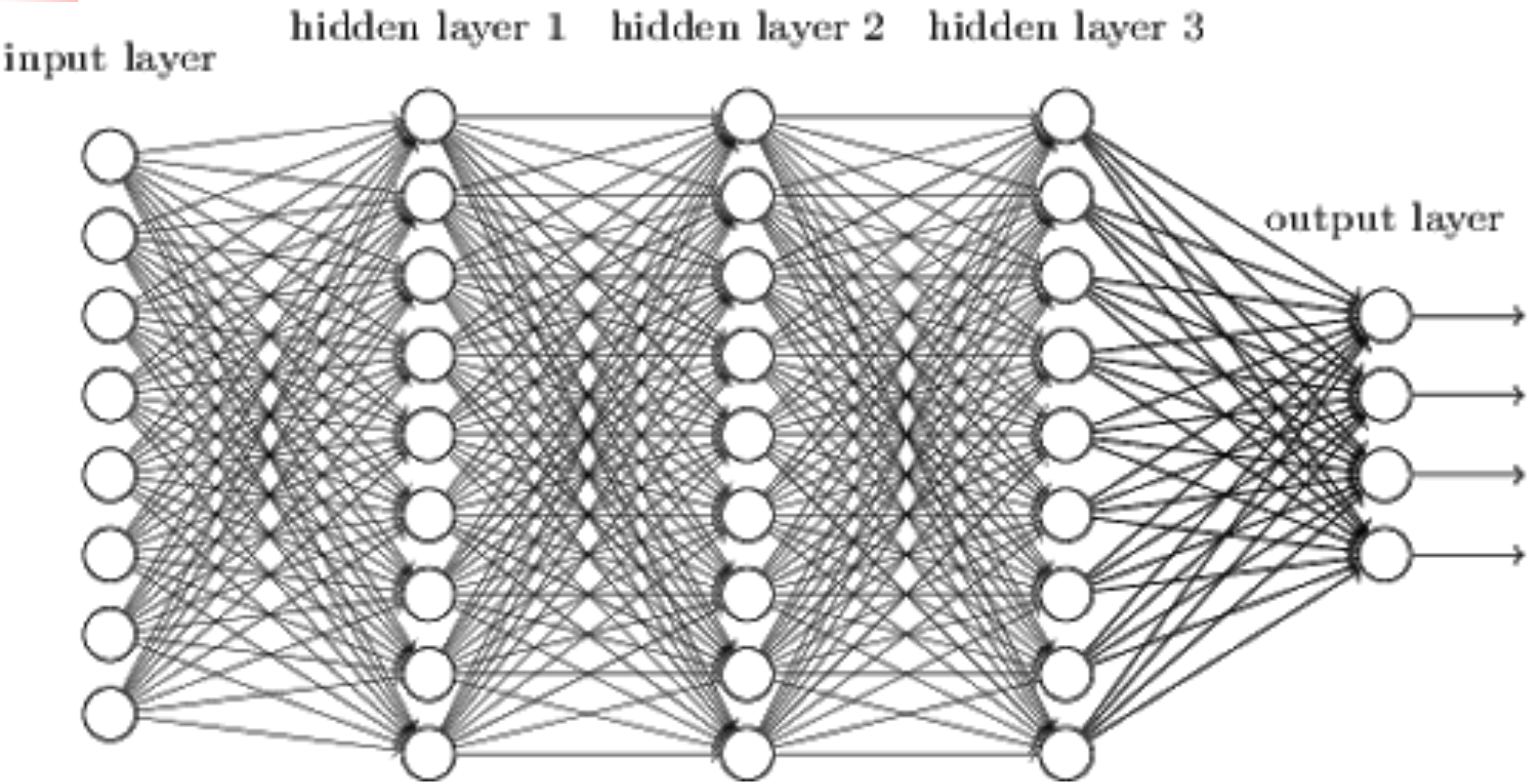
“Social” Graph (“Knows Someone”)



The Internet



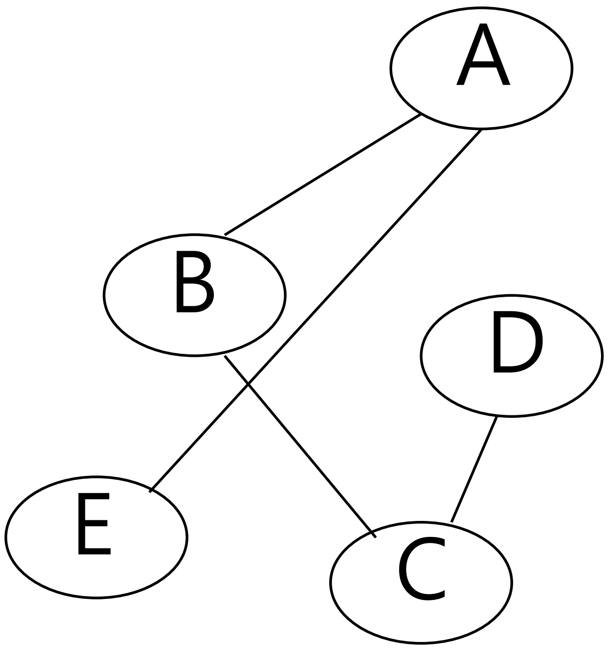
Artificial Neural Networks (for Deep Learning)



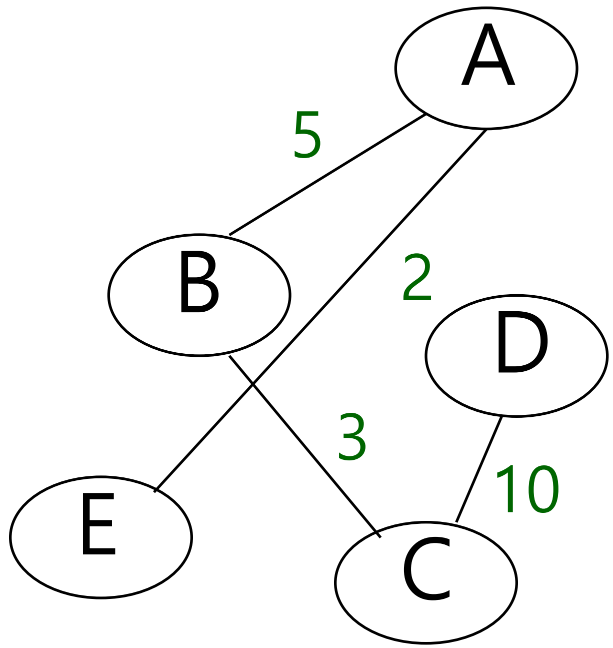
Graph: Definitions - 1

- Node (Vertex)
  - No root node, no leaf node
- Edge
  - undirected edge
    - undirected graph
  - directed edge
    - directed graph (di-graph)
    - one-way, two-way
    - in-degree, out-degree
  - weighted edge
    - (directed or undirected) weighted graph

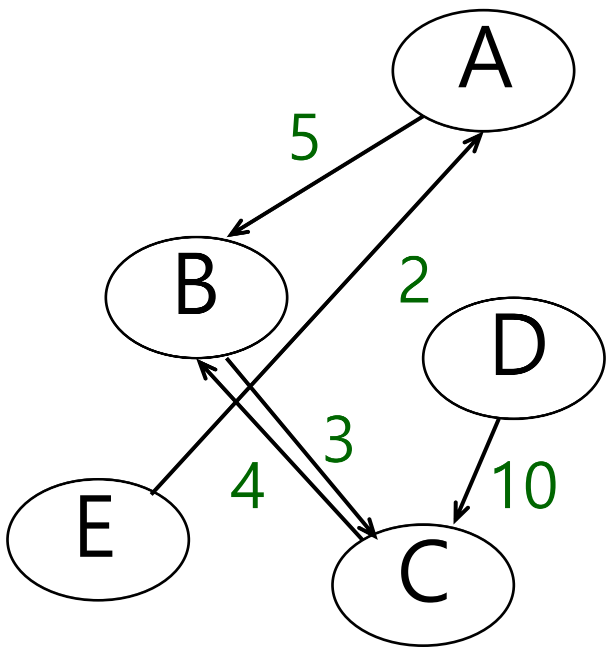
Definitions - 1: Illustrated



Definitions - 1: Illustrated (cont'd)



Definitions - 1: Illustrated (cont’d)



in-degree

A: 1

B: 2

C: 2

out-degree

A: 1

B: 1

C: 1

D: 1

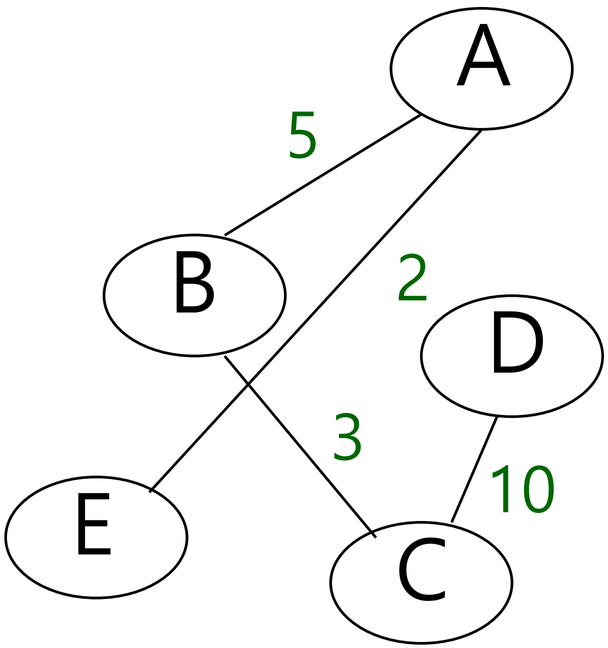
E: 1

Definitions - 2

- Subgraph
  - A part of a graph that is itself a graph
- Connected Graph
  - A graph in which any node can be reached from any other node.
- Disconnected Graph
  - A graph in which at least one node cannot be reached from some other node.
- Adjacent Nodes
  - Two nodes with a direct edge between them
- Complete (Connected) Graph
  - A special case of a connected graph
  - Every node is adjacent to every other node.

Definitions - 2: Illustrated

connected graph

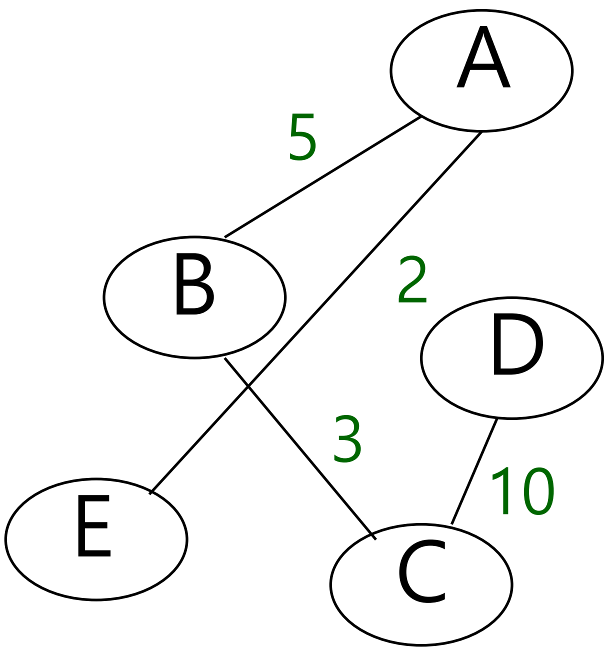


adjacent nodes

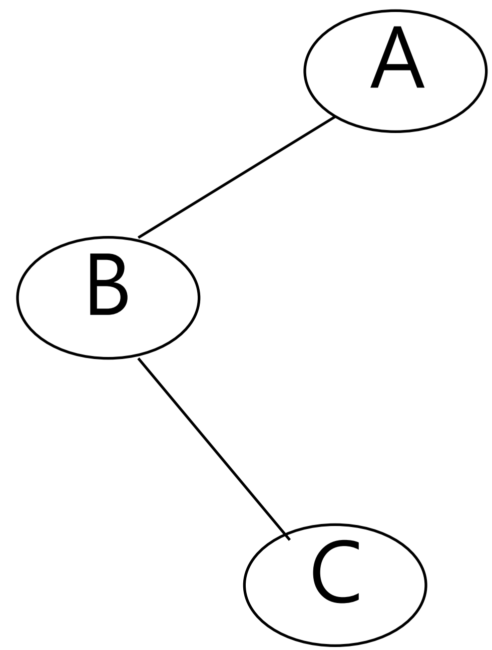
A & B
A & E
B & C
C & D

Definitions - 2: Illustrated (cont'd)

connected graph



subgraph

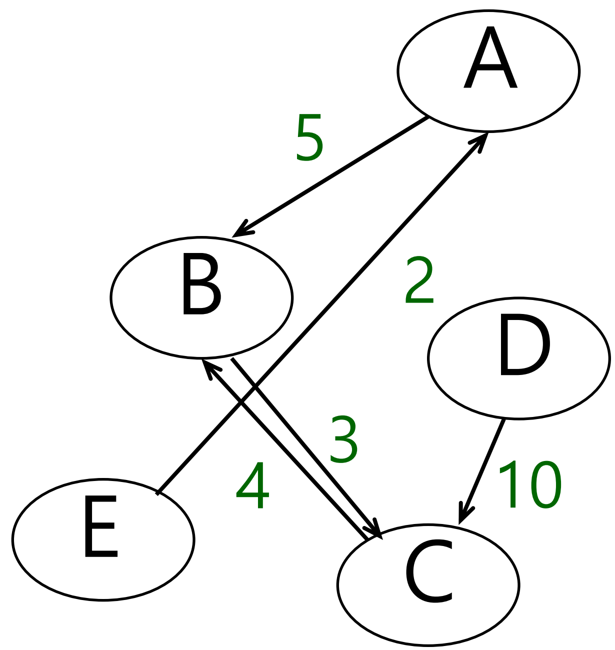


Definitions - 3

- Path (from node A to node B)
  - A sequence of edges (from node A to node B)
- Cycle
  - A path that returns to the starting node
- Cyclic Graph
  - A graph with a cycle
- Acyclic Graph
  - A graph with no cycle

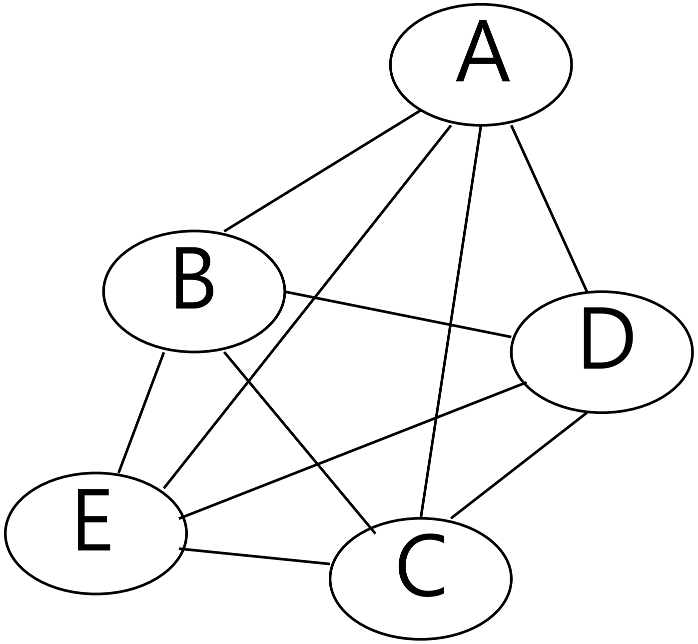
Definitions - 2: Illustrated (cont'd)

disconnected graph



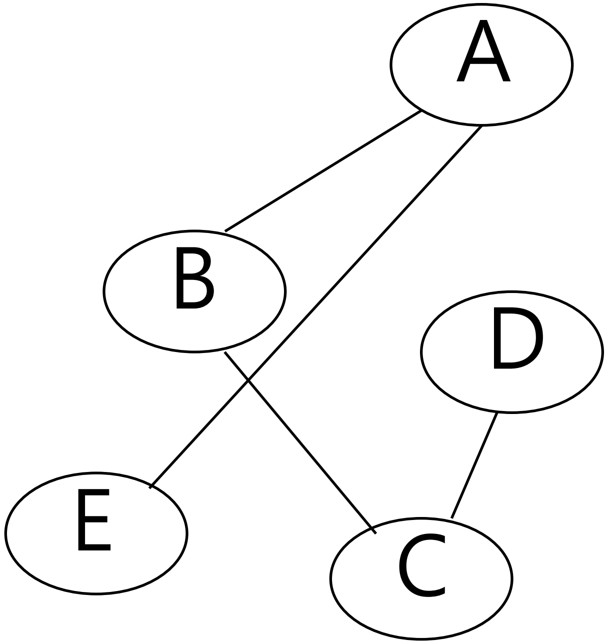
Definitions - 2: Illustrated (cont'd)

complete graph

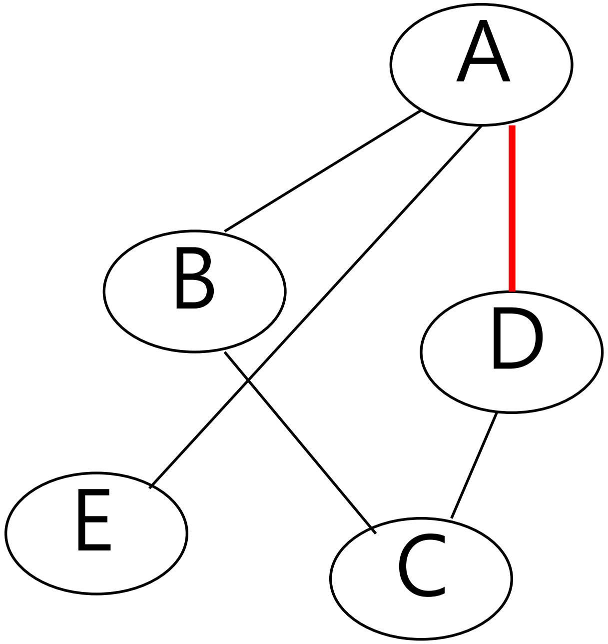


Definitions - 3: Illustrated

acyclic graph



cyclic graph



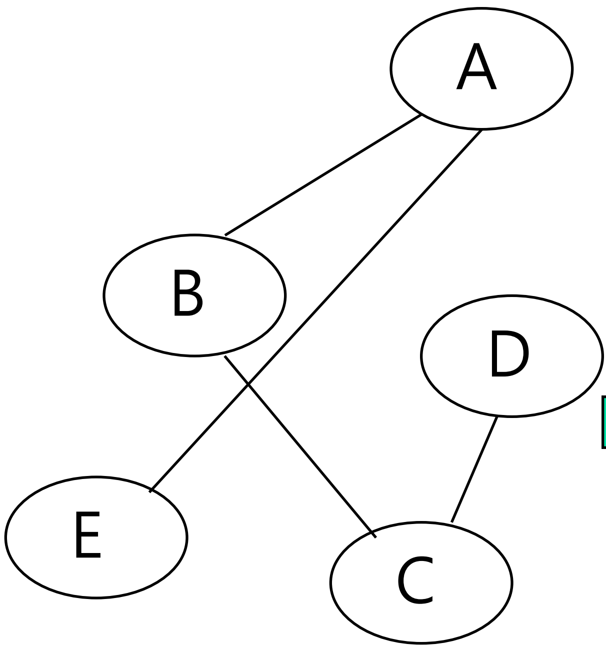
Graph Algorithms

- Graph representation
- Graph traversal
- Spanning tree
- Minimum spanning tree
- Minimum spanning tree algorithms
- Transitive closure algorithms

Representation of a Graph

- Adjacency Matrix
  - Create an (n x n) 2-dimensional matrix
    - n is the number of nodes in the graph.
  - A (1 or 0) entry for each (row-column) node pair
    - 1 if the pair is adjacent; 0 otherwise
- Adjacency Lists
  - For each node, create a linked list of adjacent nodes

Representation of a Graph: Example 1

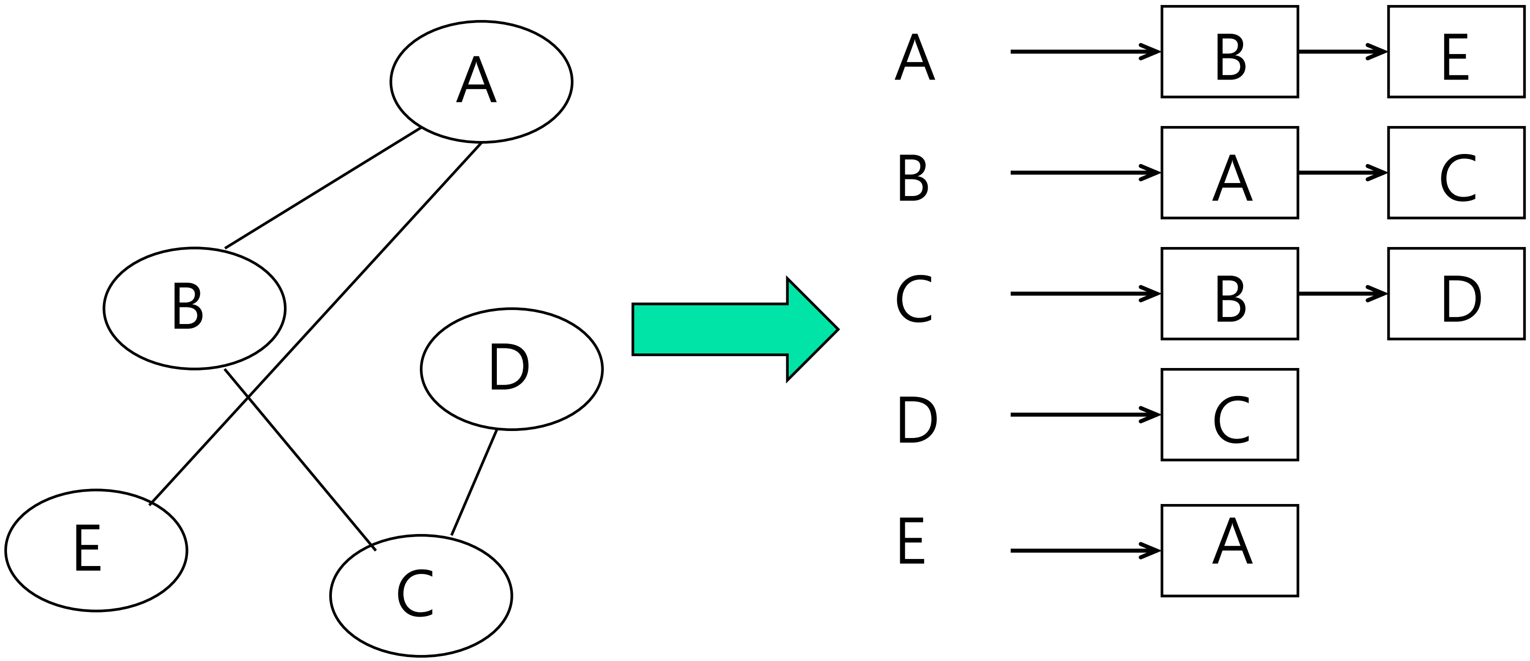


adjacency matrix

	A	B	C	D	E
A		1			1
B	1		1		
C		1		1	
D			1		
E	1				

Representation of a Graph: Example 2

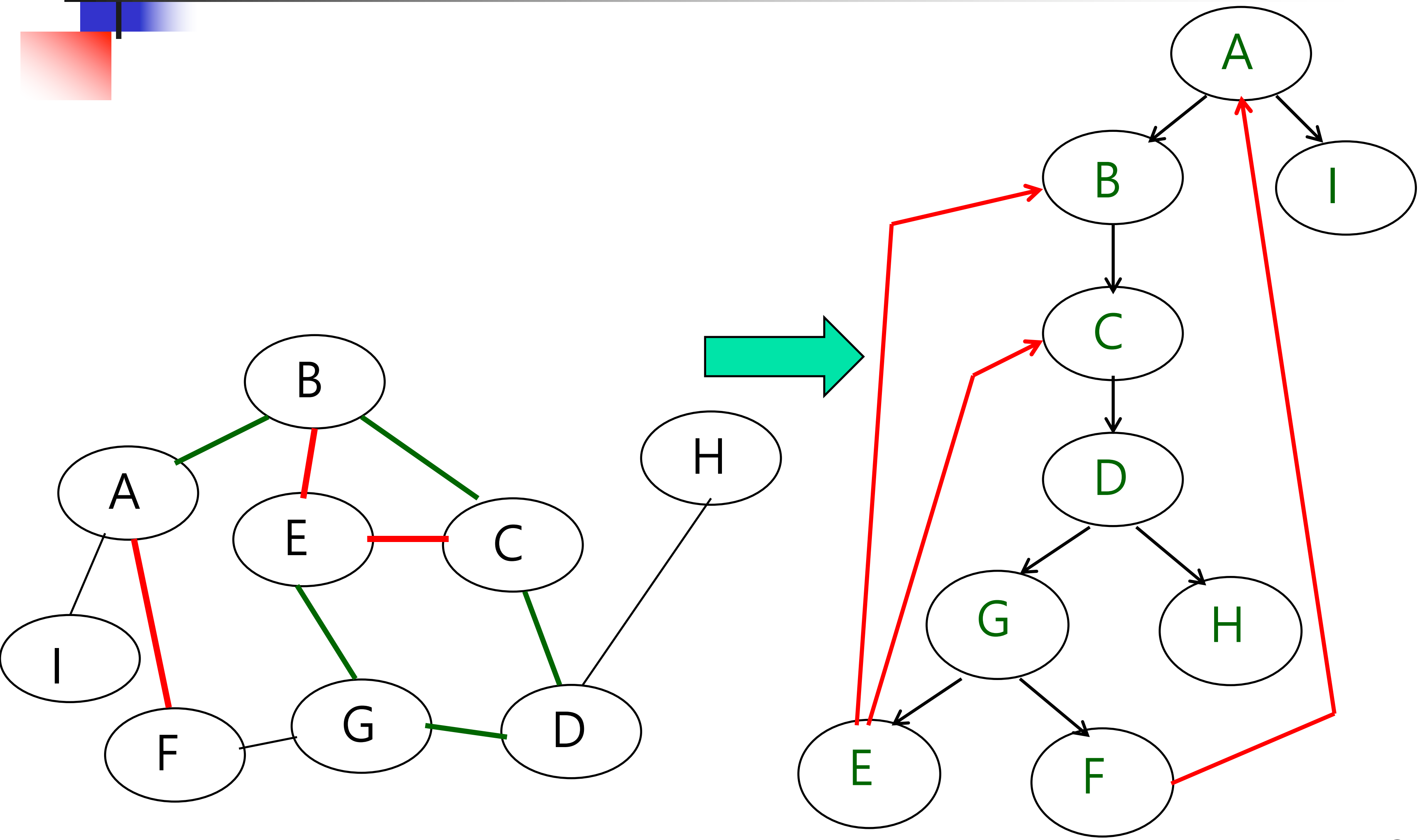
adjacency lists



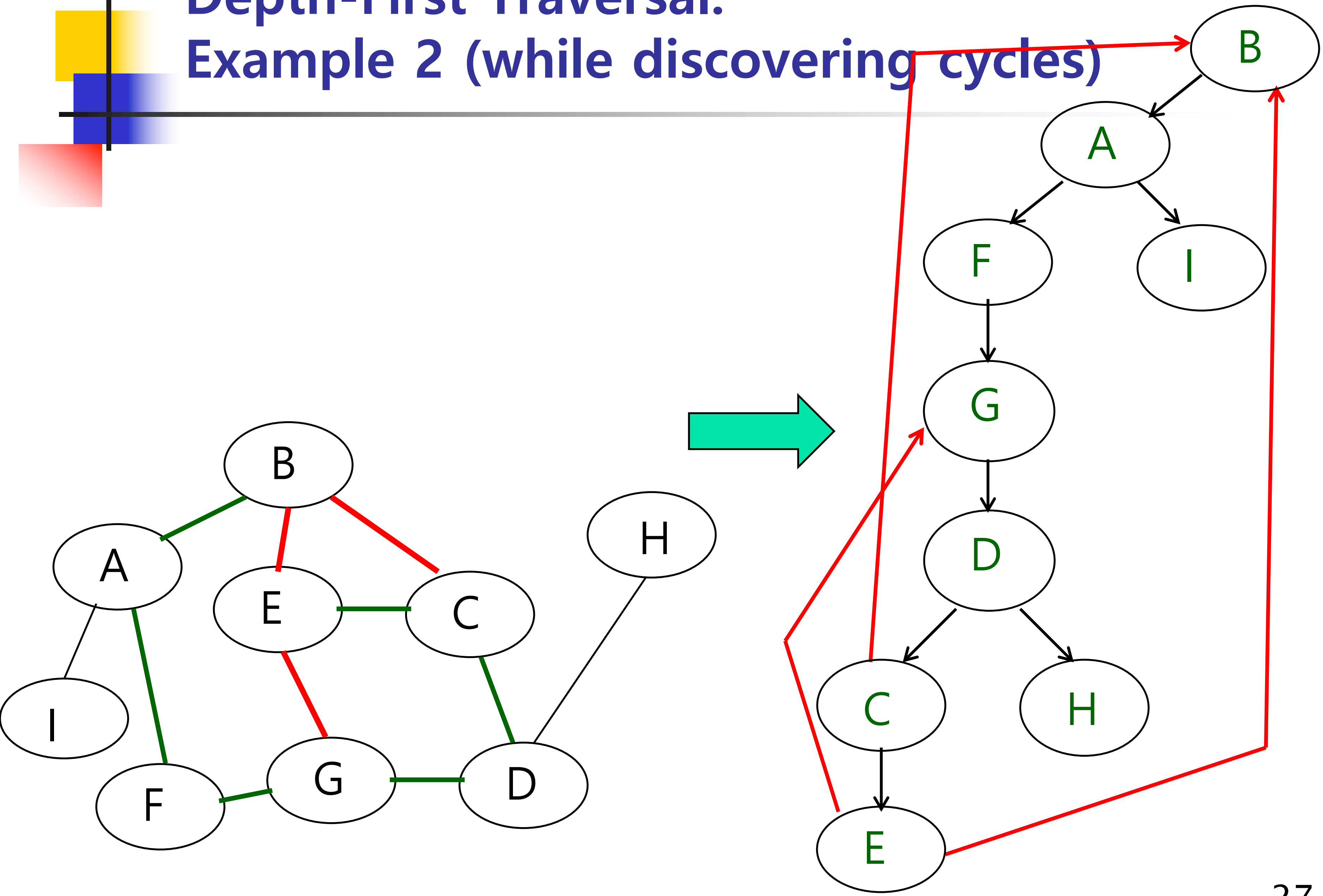
Graph Traversal

- Depth-First Traversal
  - From a node, visit one adjacent node recursively, until there is no new adjacent node.
  - Then backtrack, and visit the next adjacent node recursively.
  - Finish when every adjacent node of the initial node has been recursively traversed.
- Breadth-First Traversal
  - From a node, visit adjacent nodes one at a time.
  - Then visit all unvisited adjacent nodes of each of the adjacent nodes of the original node one at a time.
  - Finish when all nodes on each of the paths from the original node have been visited.

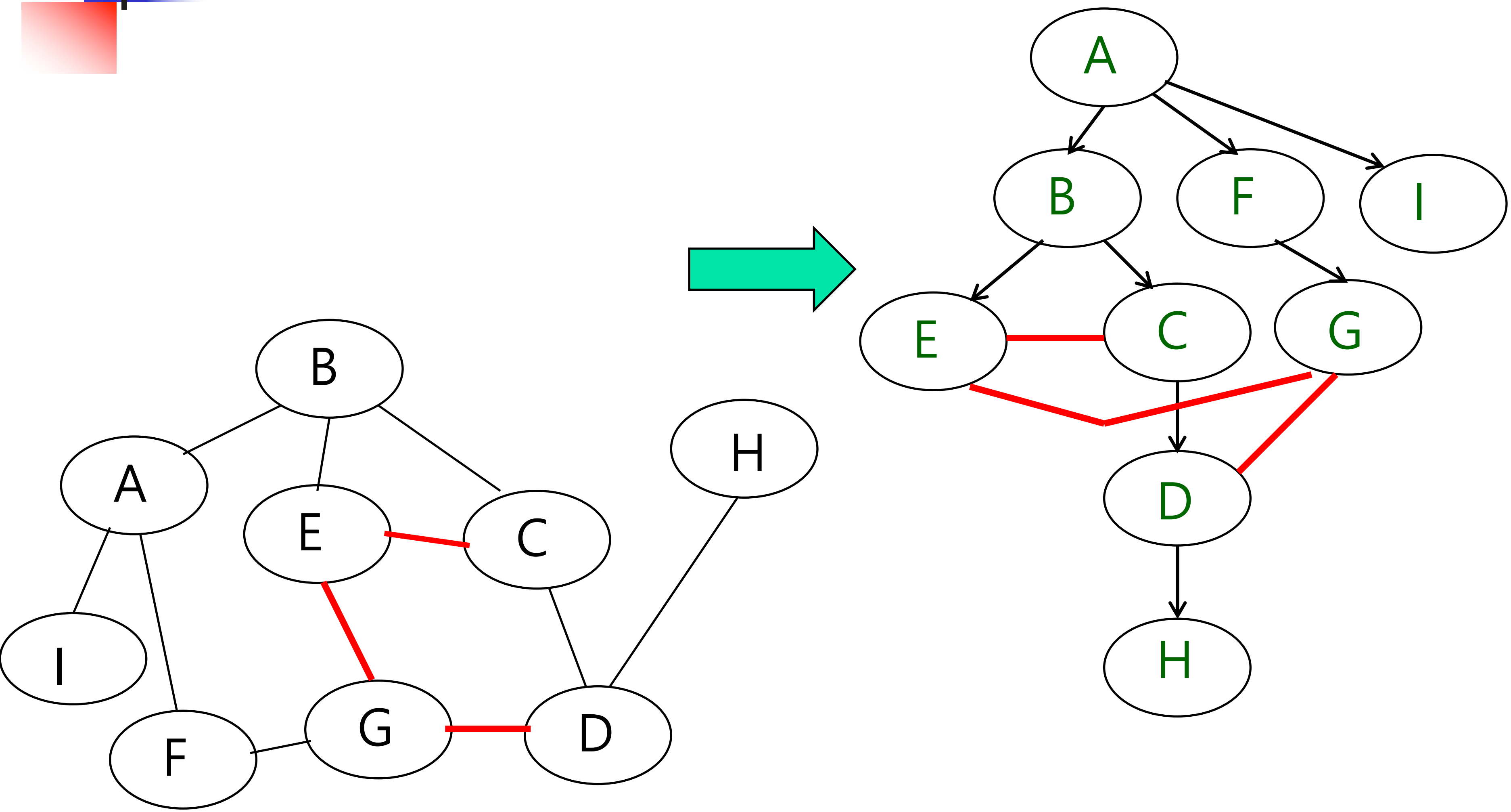
Depth-First Traversal: Example 1 (while discovering cycles)



Depth-First Traversal: Example 2 (while discovering cycles)



Breadth-First Traversal: Example (while discovering cycles)

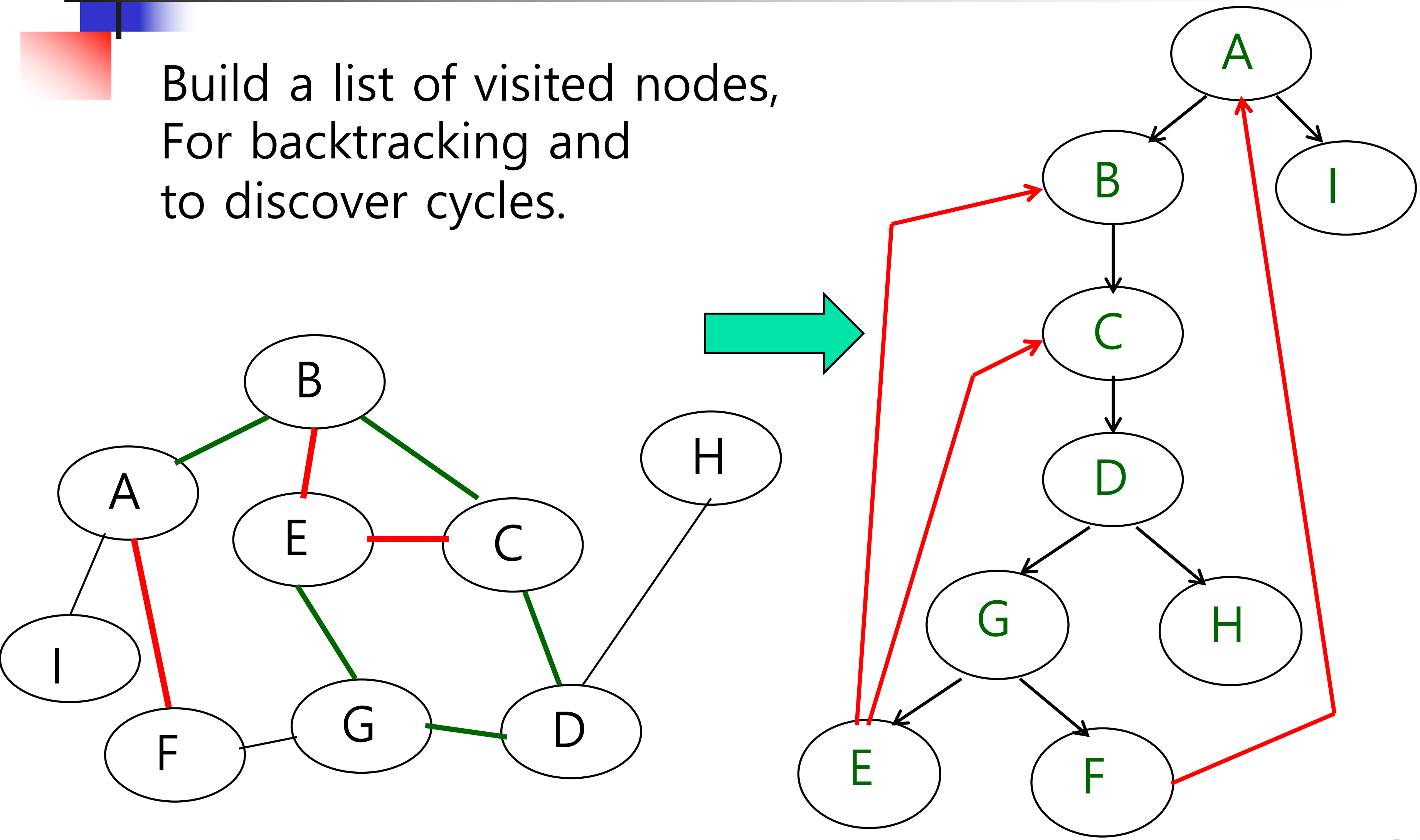


Spanning Tree

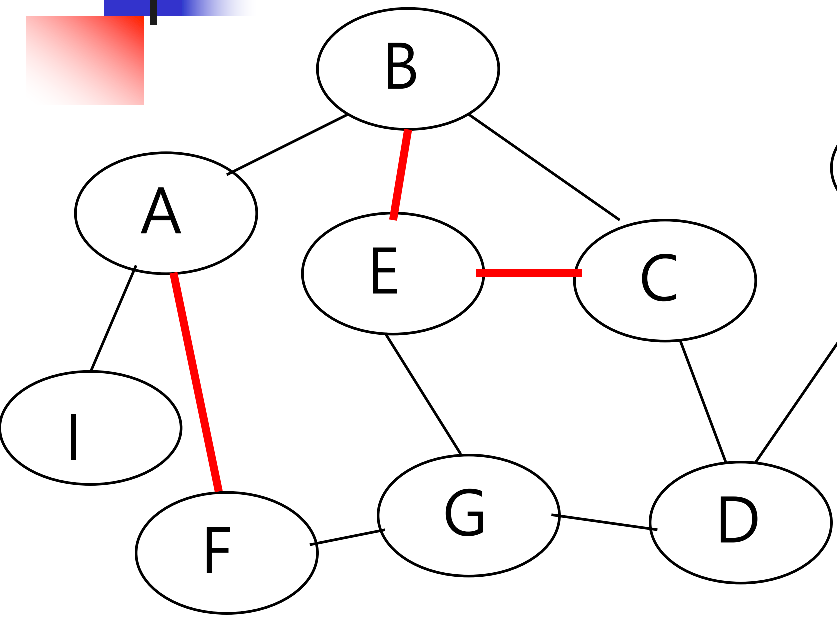
- Obtained from a connected graph of N nodes
  - Via a depth-first traversal or breadth-first traversal
  - A tree of N nodes and N-1 edges
  - Cycles are removed from the graph.
- More than one spanning tree may be obtained from the same connected graph.

Depth-First Traversal: Example 1

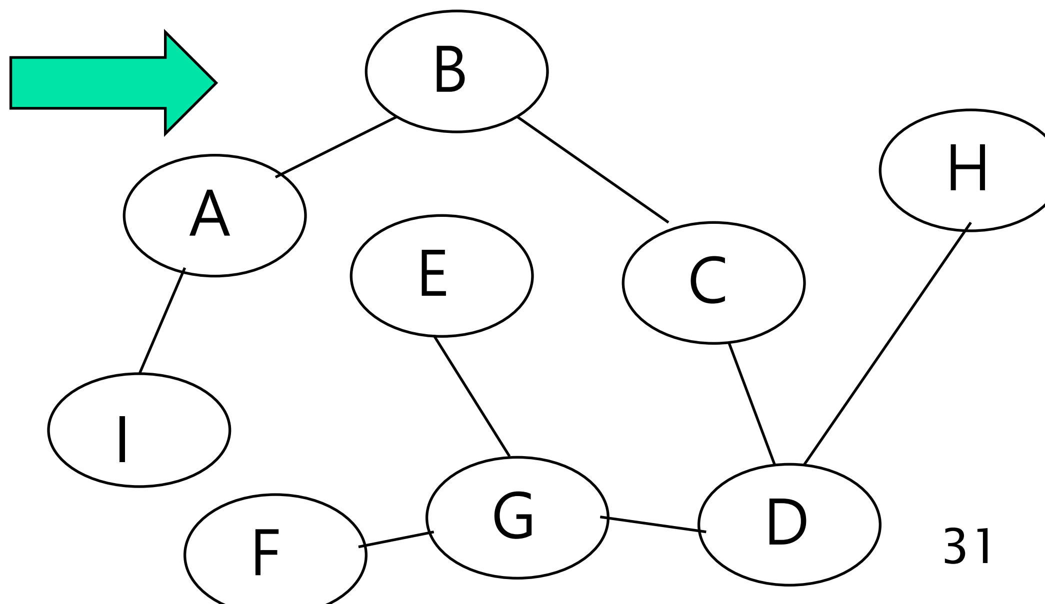
Build a list of visited nodes,
For backtracking and
to discover cycles.



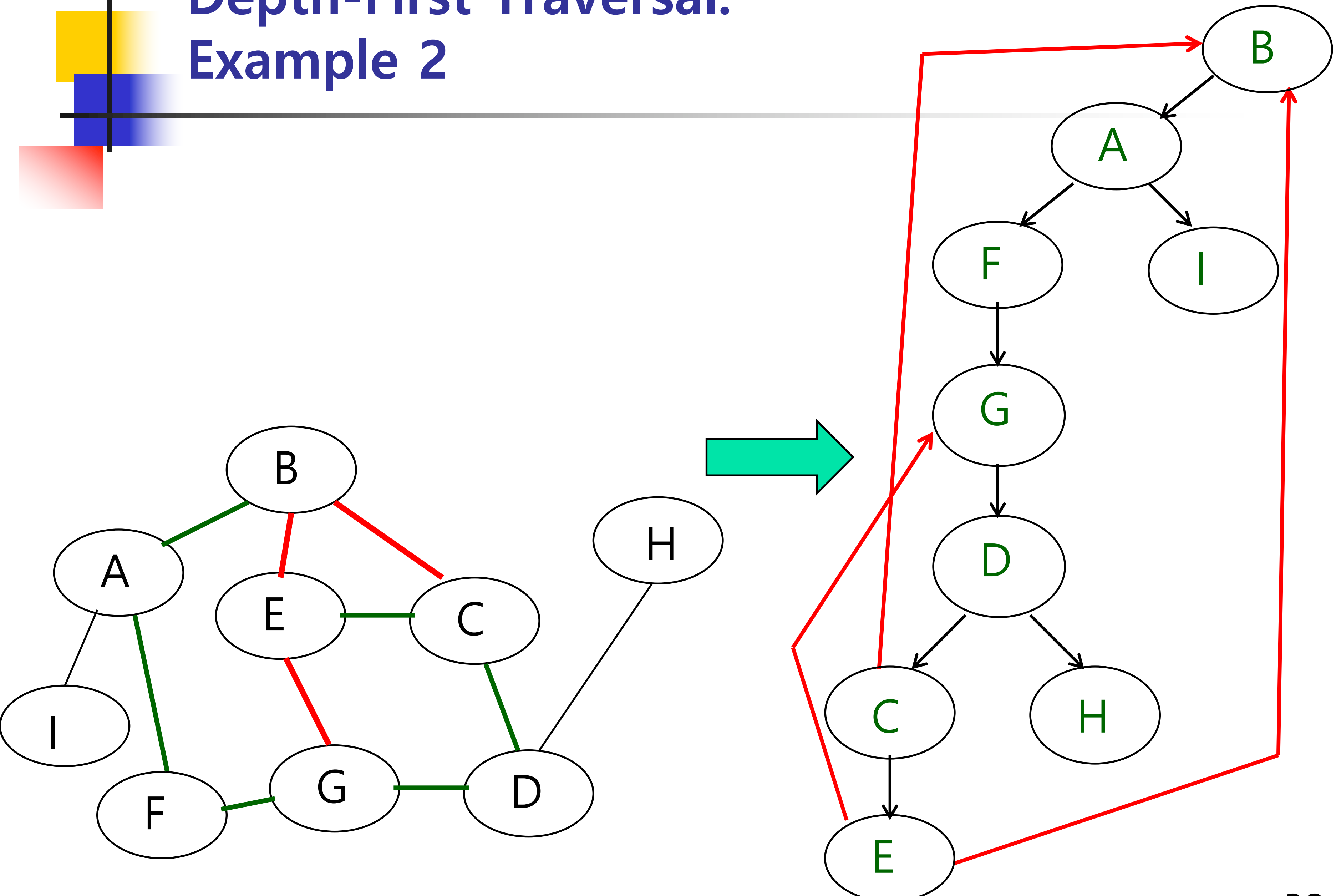
Depth-First Traversal: Example 1 (cont'd)



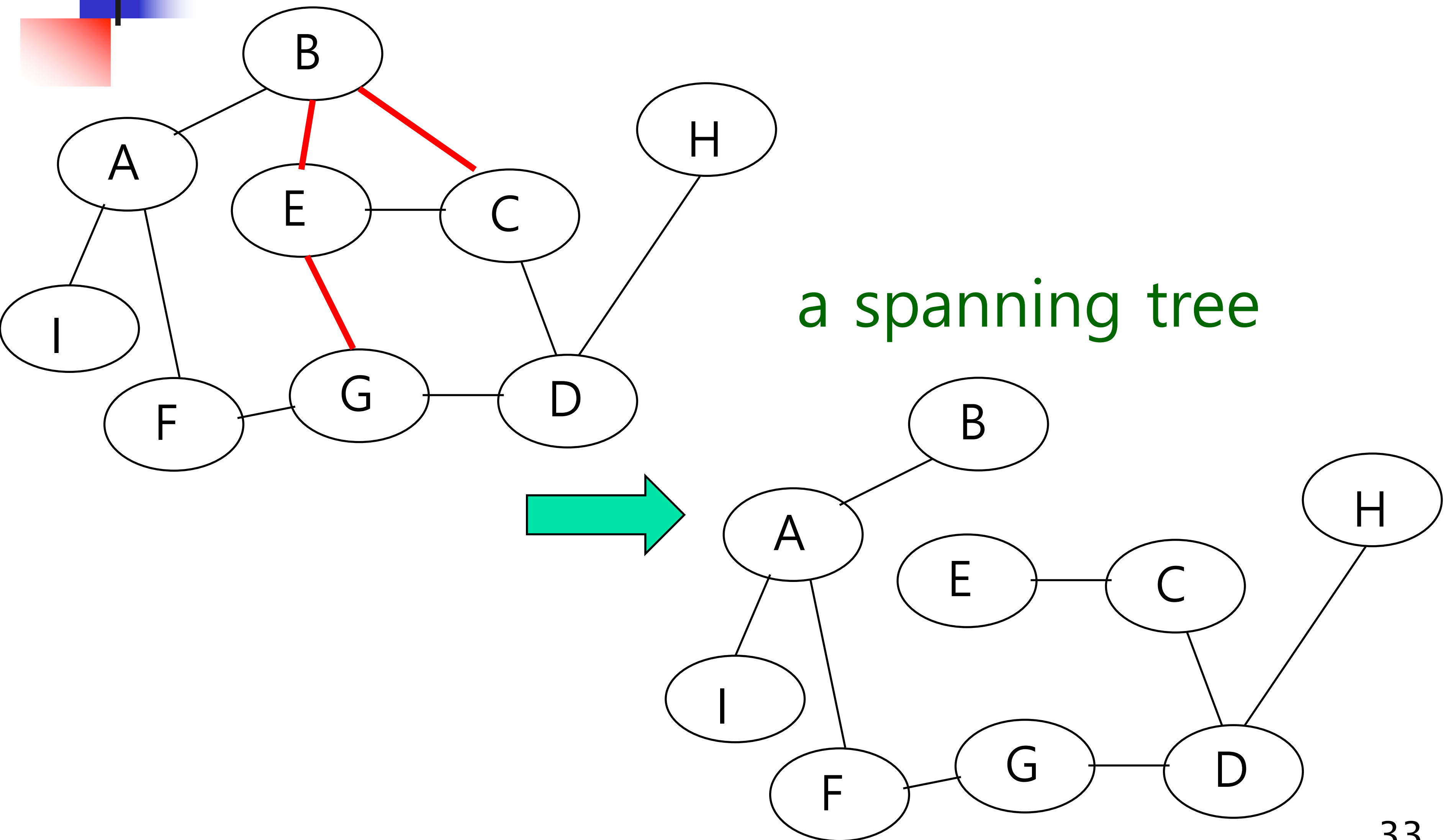
a spanning tree



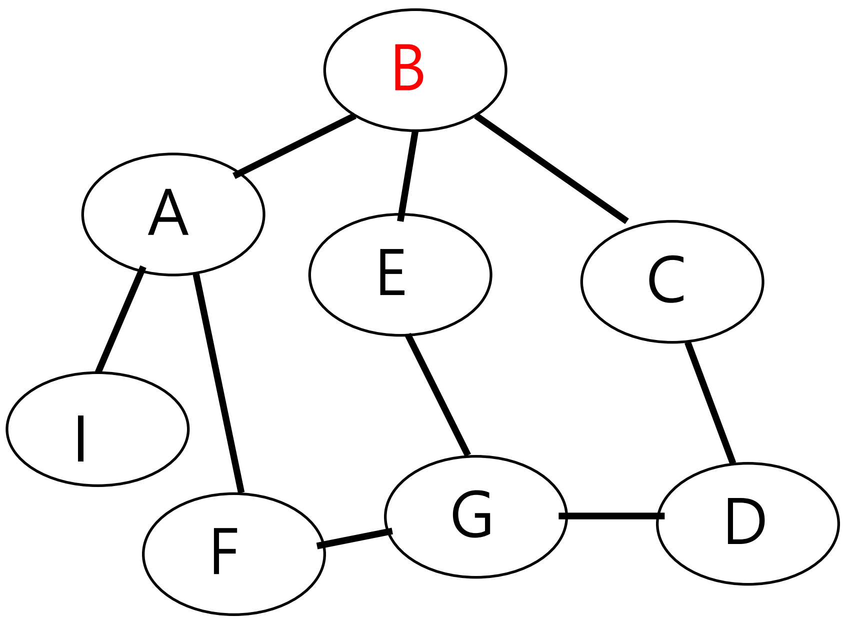
Depth-First Traversal: Example 2



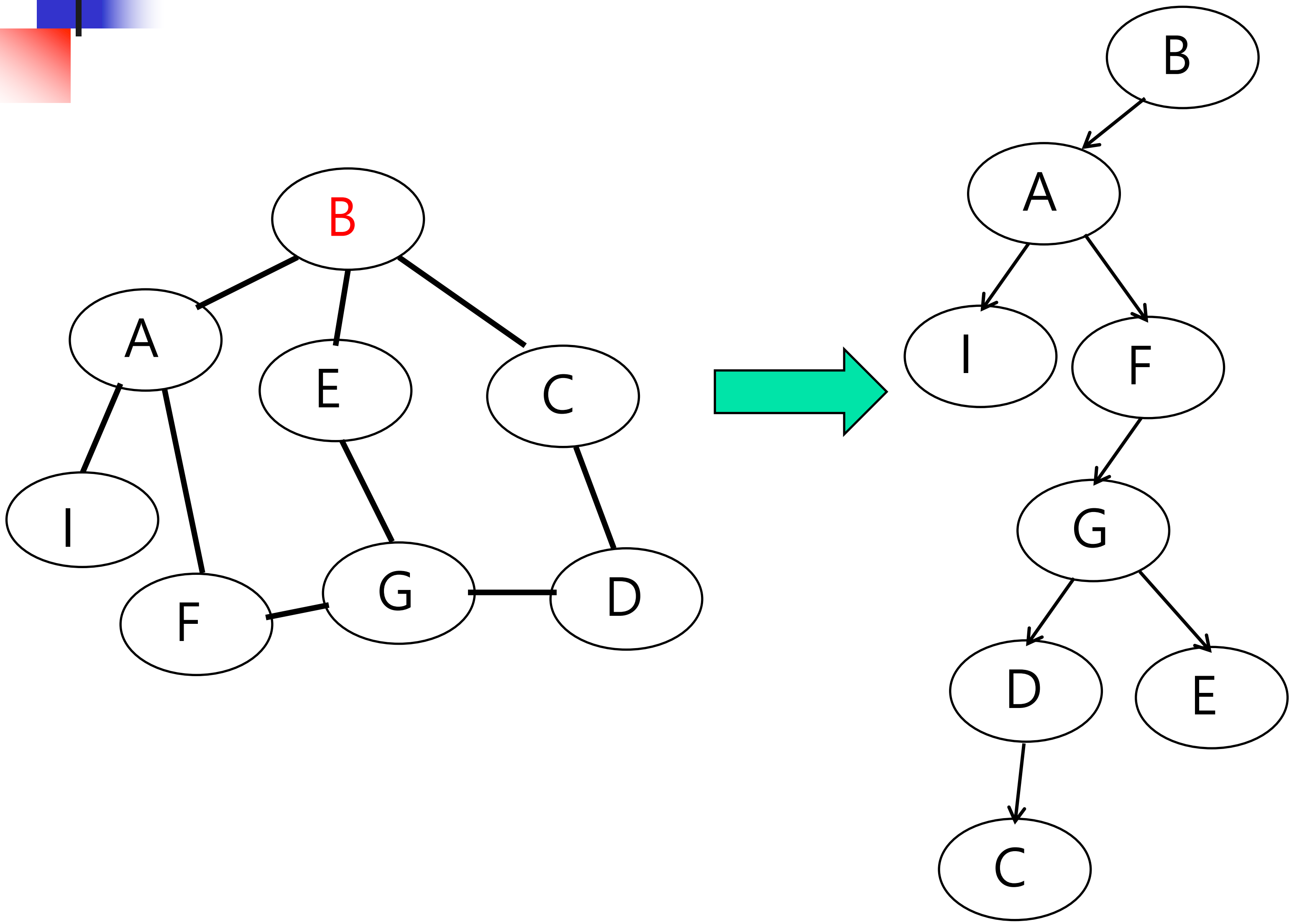
Depth-First Traversal: Example 2 (cont'd)



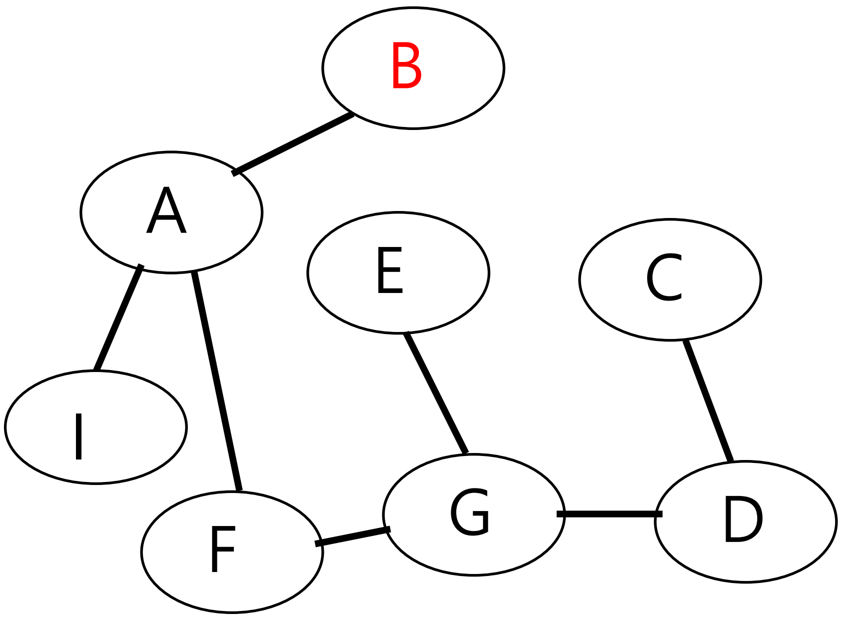
Exercise 1: Obtain One Spanning Tree through a Depth-First Traversal



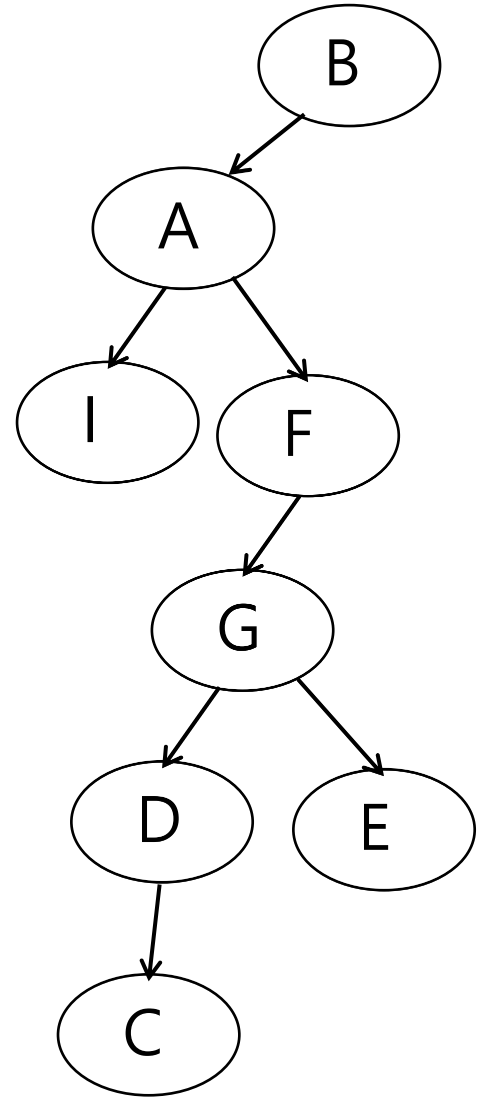
Exercise 1: Solution



Exercise 1: Solution

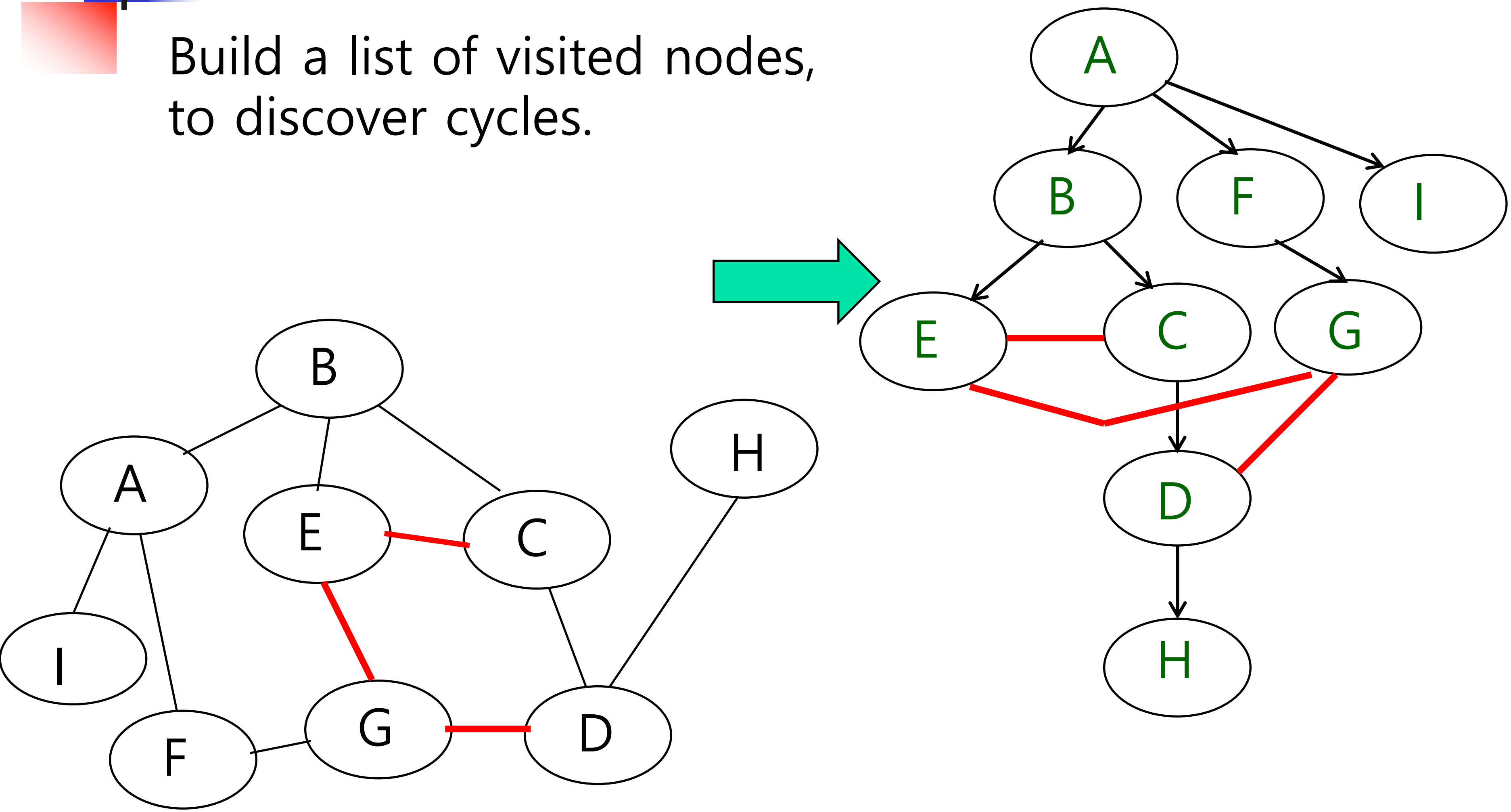


a spanning tree

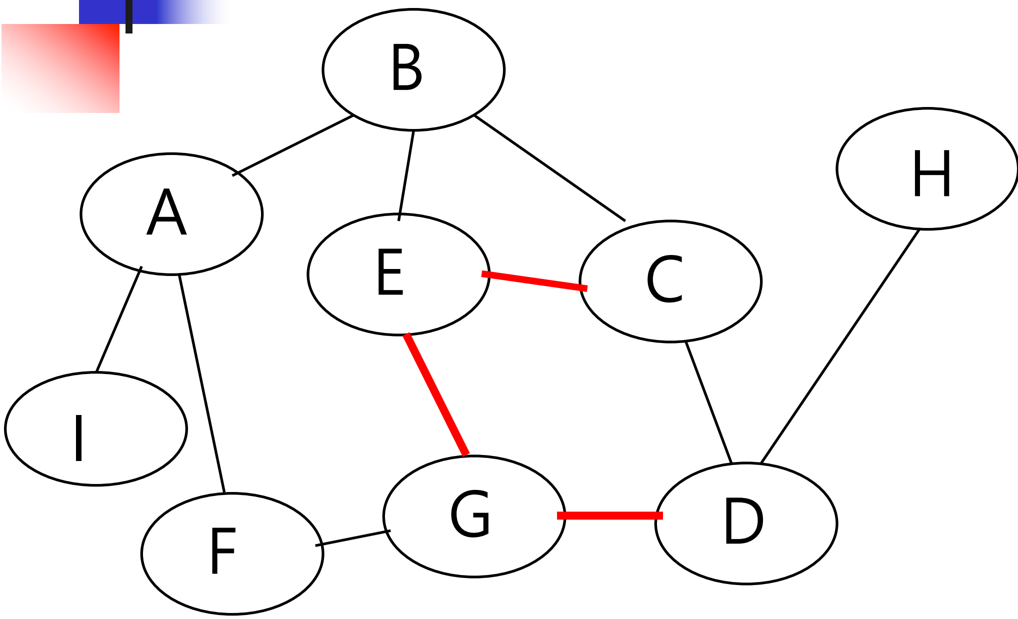


Breadth-First Traversal: Example

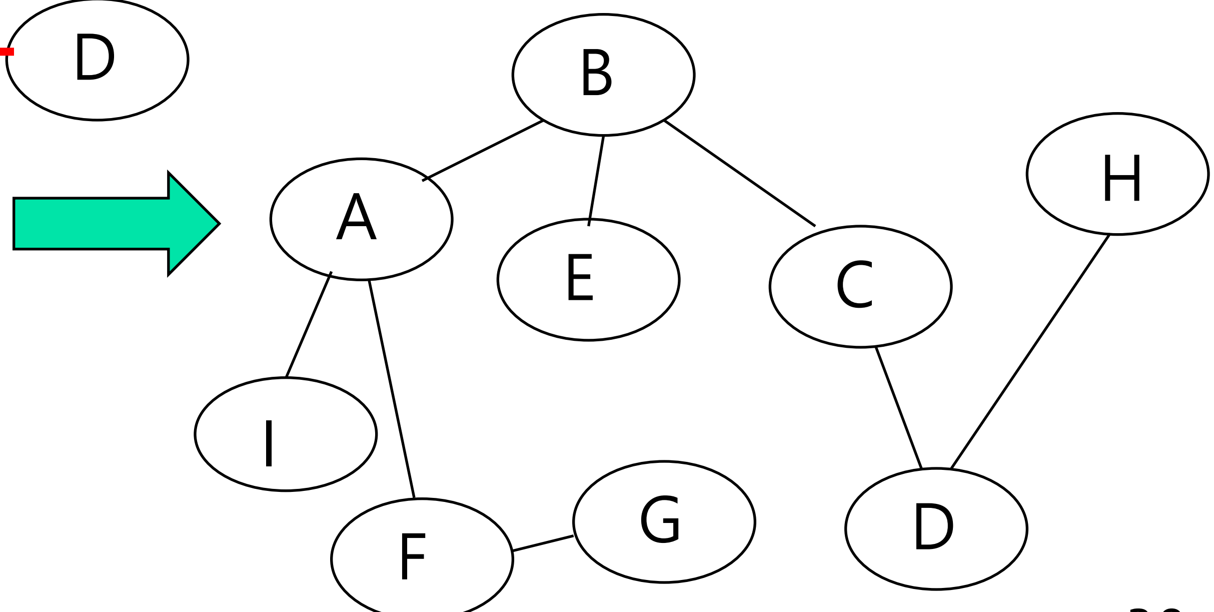
- Build a list of visited nodes, to discover cycles.



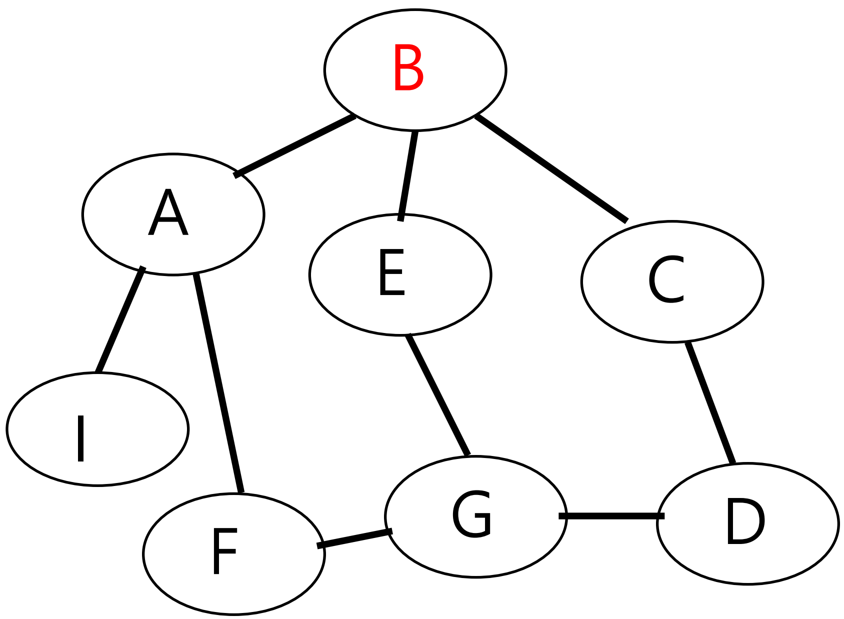
Breadth-First Traversal: Example (cont'd)



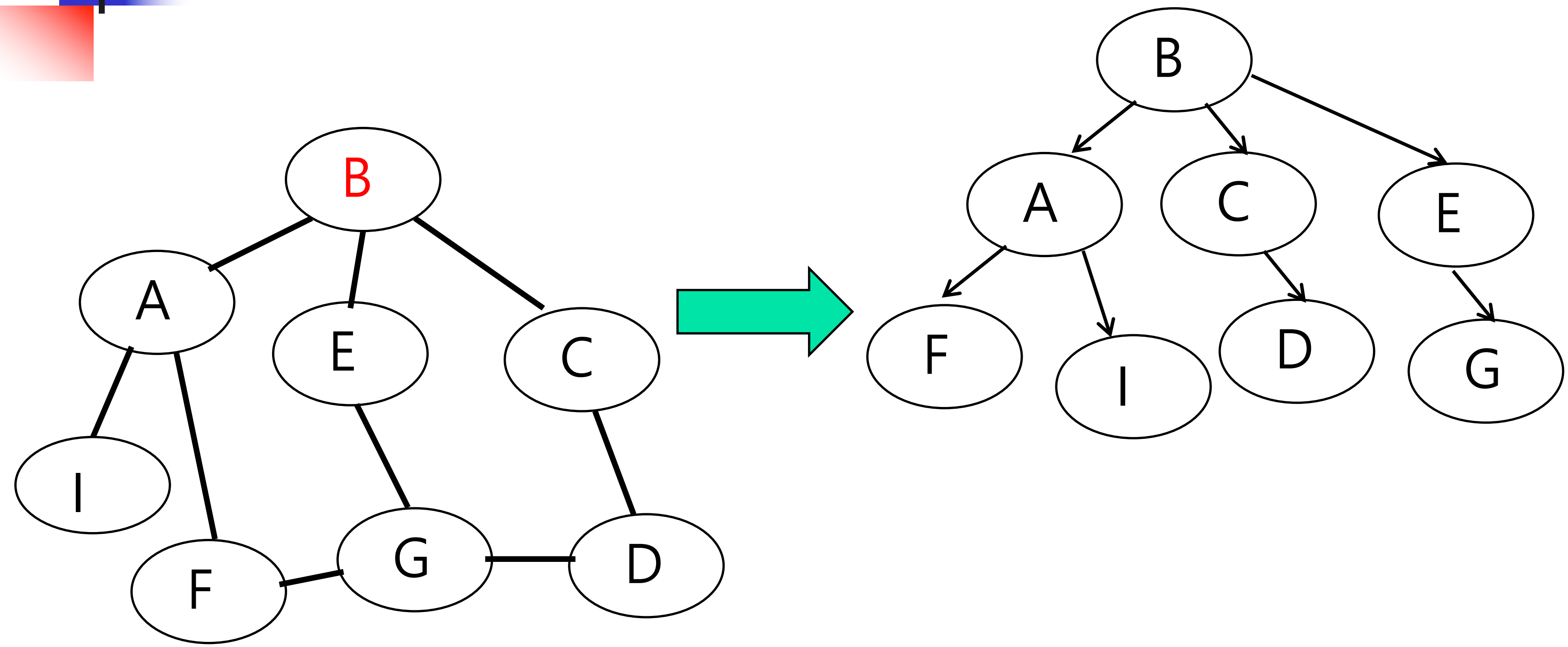
a spanning tree



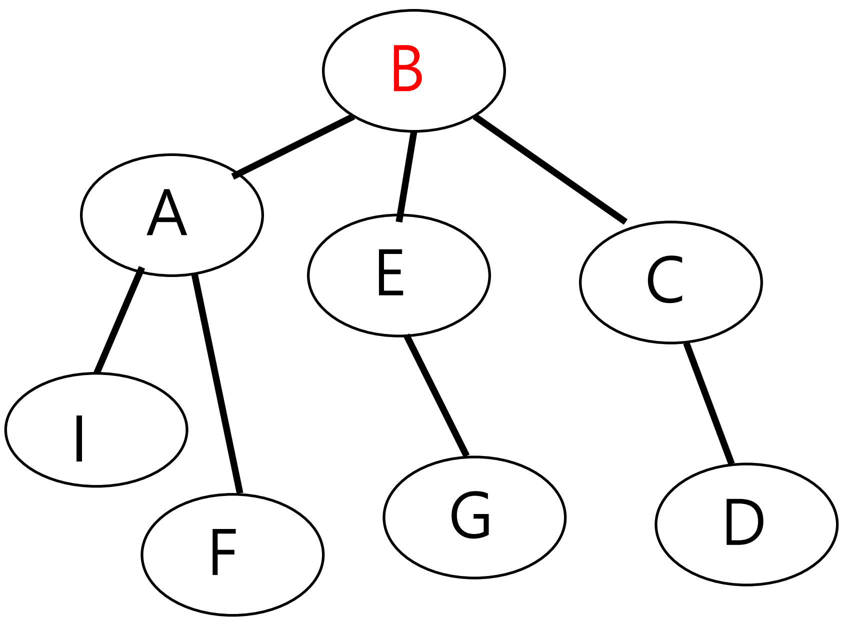
Exercise 2: Obtain One Spanning Tree through a Breadth-First Traversal

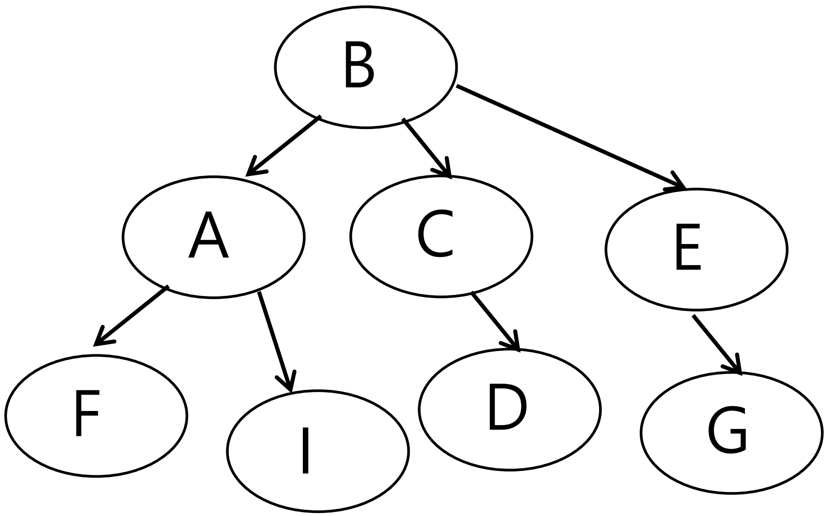


Exercise 2: Solution



Exercise 2: Solution





a spanning tree

Summary Definition of a Graph

- A graph is a linked list of data, where a data item is linked to other data items by a certain relationship.
- A search for a data item on a graph proceeds from one data item to another data item that satisfies a certain relationship. The search from a data item may return to itself.
- The network of a graph is a convenient visualization of the relationships among the data items.

Minimum Spanning Tree

- Obtained from a weighted connected graph of N nodes
- A spanning tree with A minimum total weight of all the edges.
- In general not unique
- Used to model and solve some “minimum-cost” problems
  - (e.g.) minimum-cost construction of subway system routes

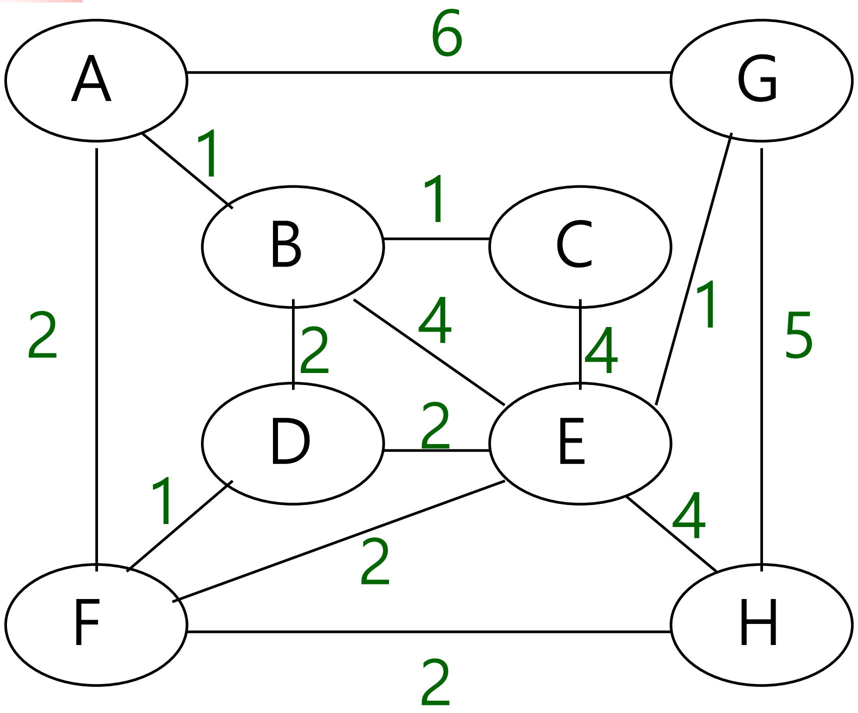
Minimum Spanning Tree

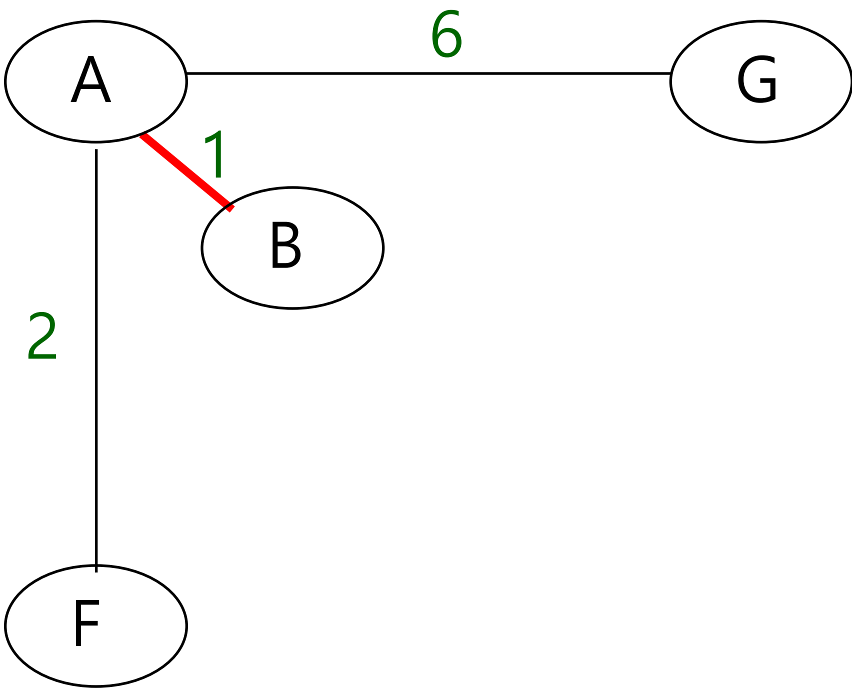
- Three well-known algorithms
  - Prim’s algorithm
  - Kruskal’s algorithm
  - Sollin’s algorithm

Prim's Algorithm

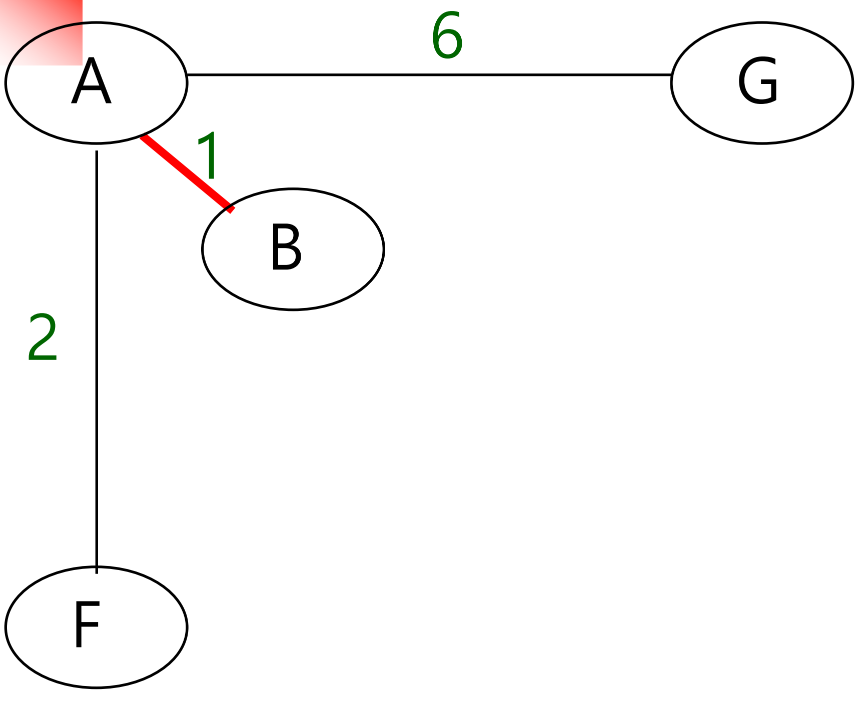
- Start from a node of an undirected graph.
- Examine all adjacent nodes.
- Pick a minimum-weight edge
  - (and insert it into a list of visited nodes).
- Continue from the selected adjacent node.
  - Examine all edges not selected.
  - Examine all edges to new adjacent nodes.
  - Pick a minimum-weight edge, avoiding cycles.
    - (And insert it into a list of visited nodes)
- Continue until all nodes of the graph have been included (in the list of visited nodes(and all edges have been considered at least once.

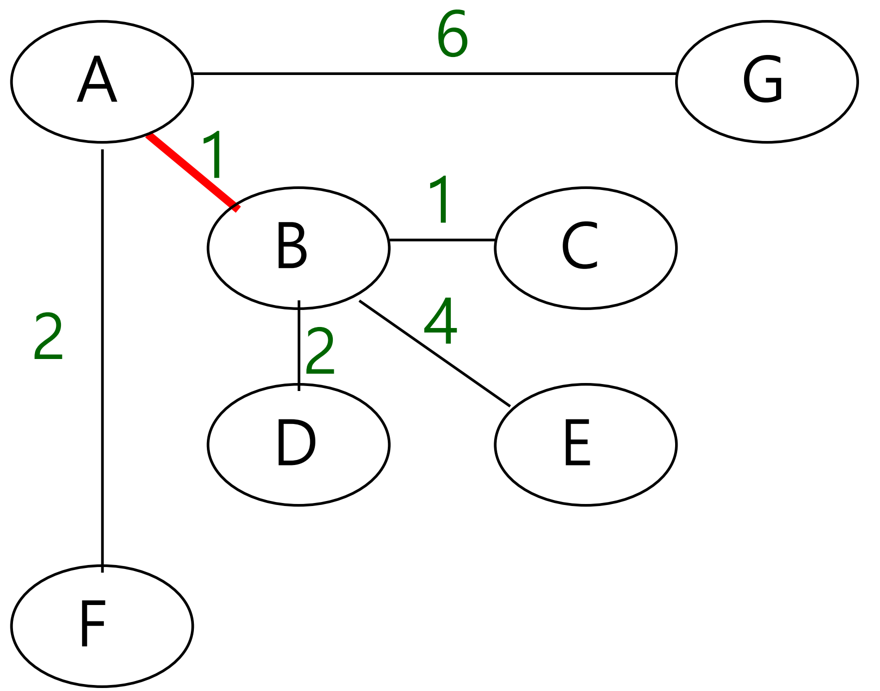
Prim's Algorithm: Illustrated



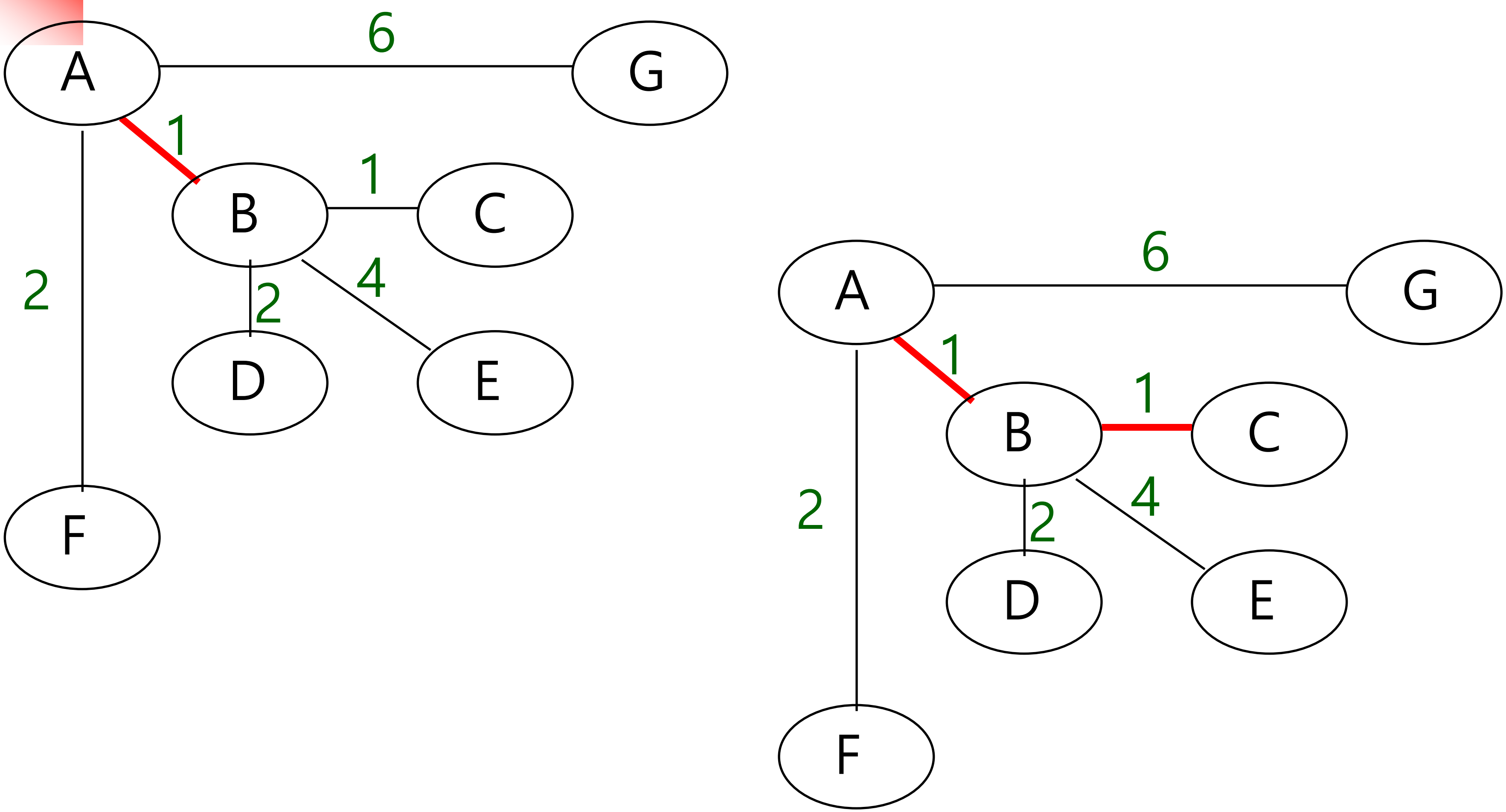


Prim's Algorithm: Illustrated (cont'd)

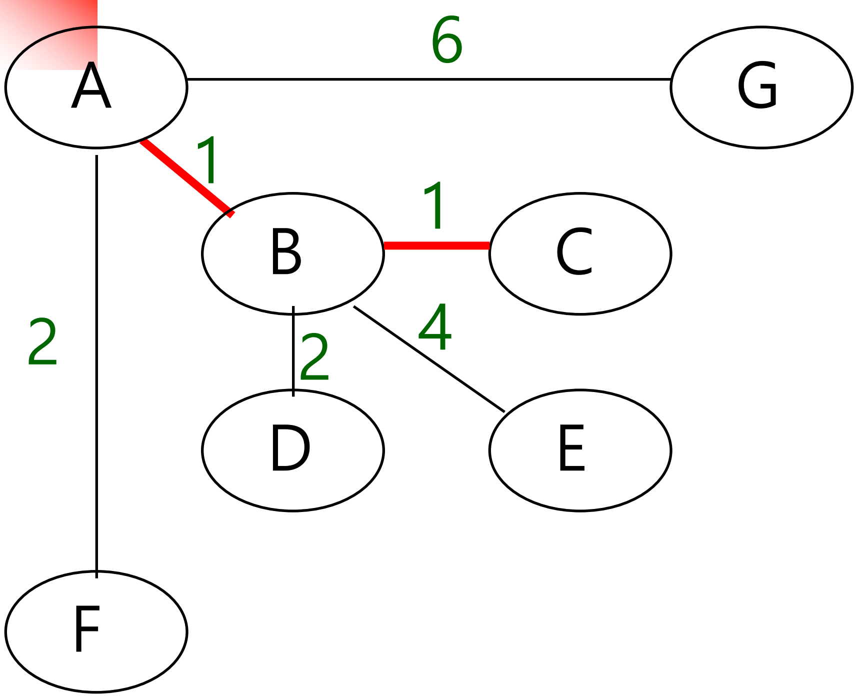


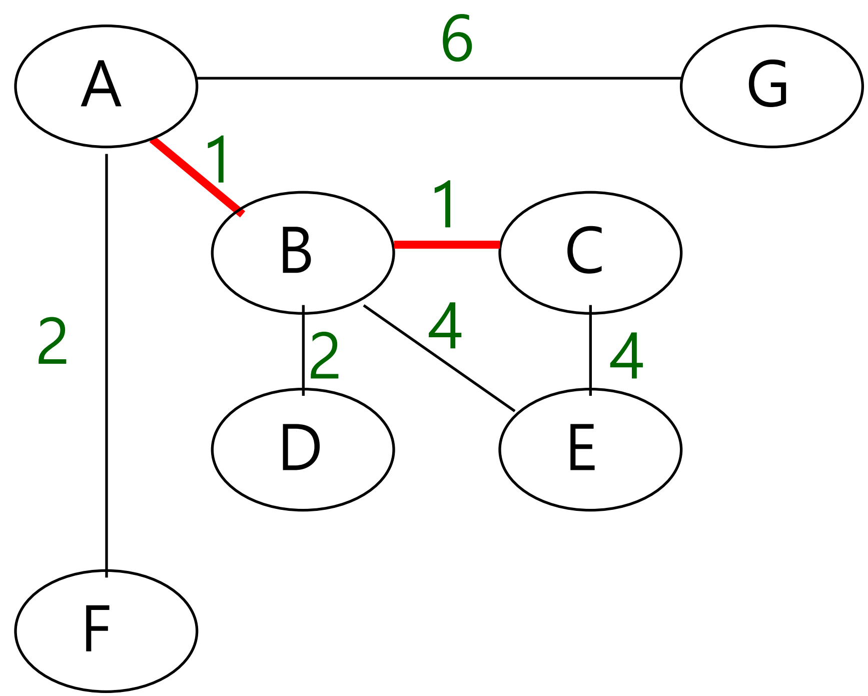


Prim's Algorithm: Illustrated (cont'd)

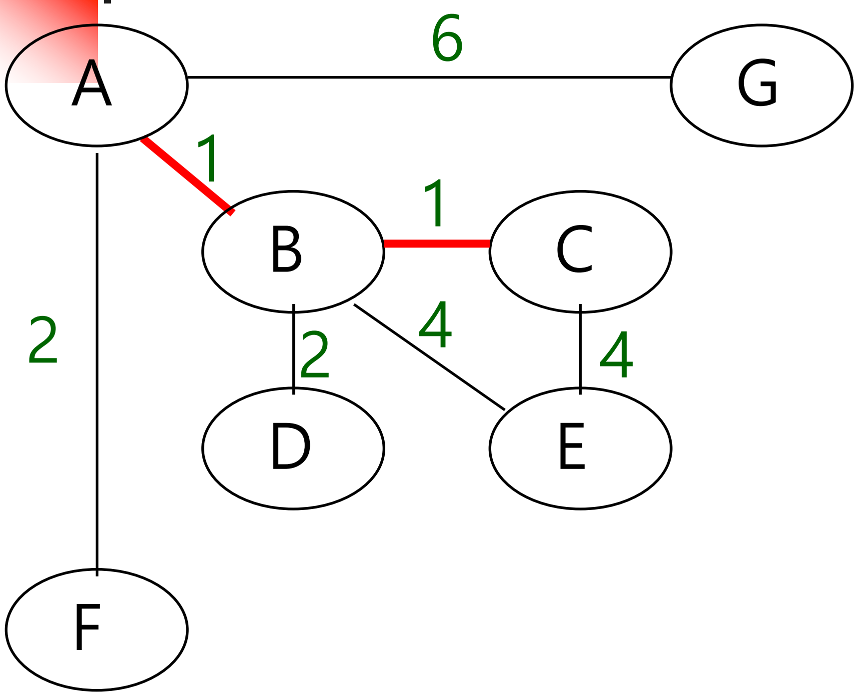


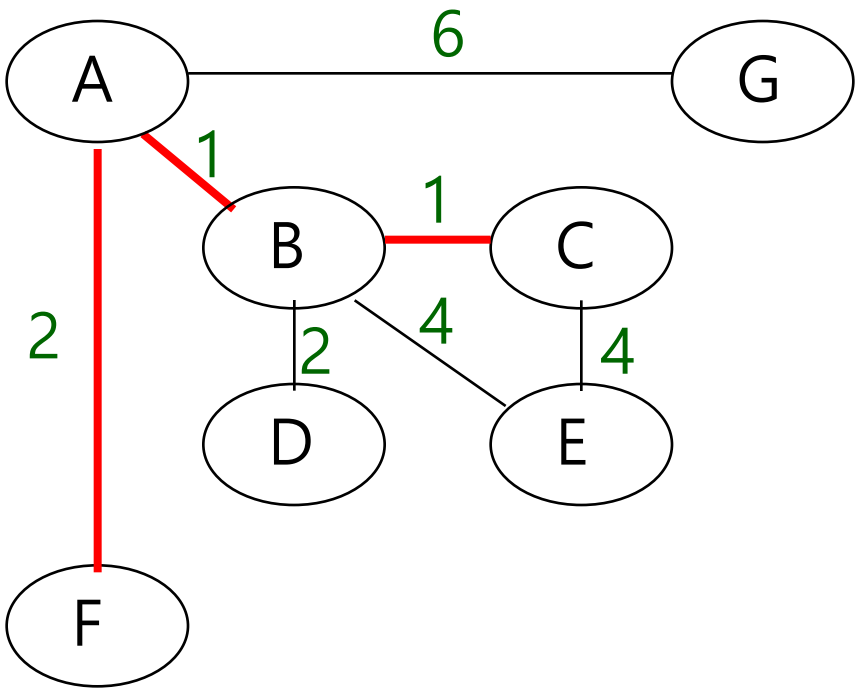
Prim's Algorithm: Illustrated (cont'd)



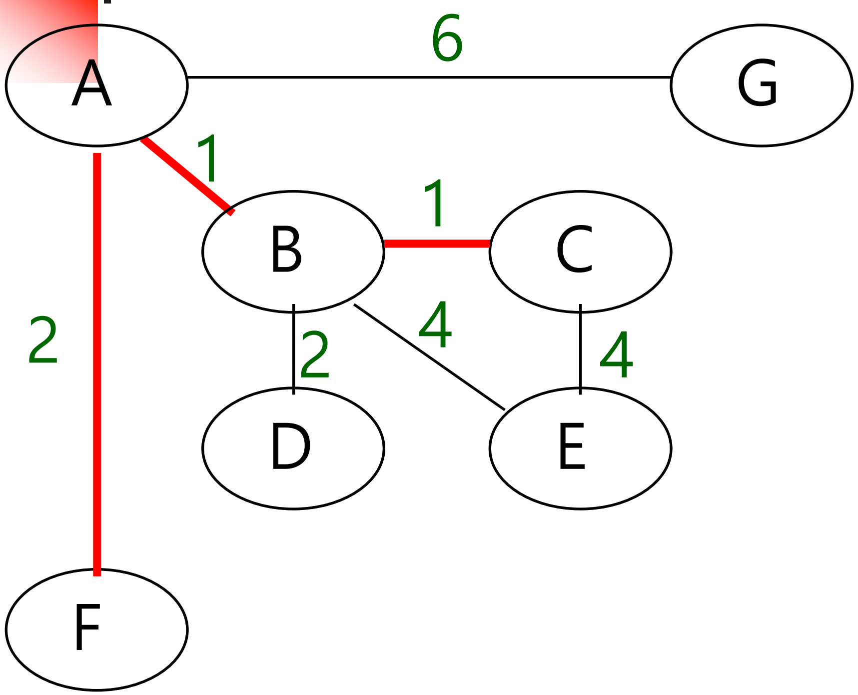


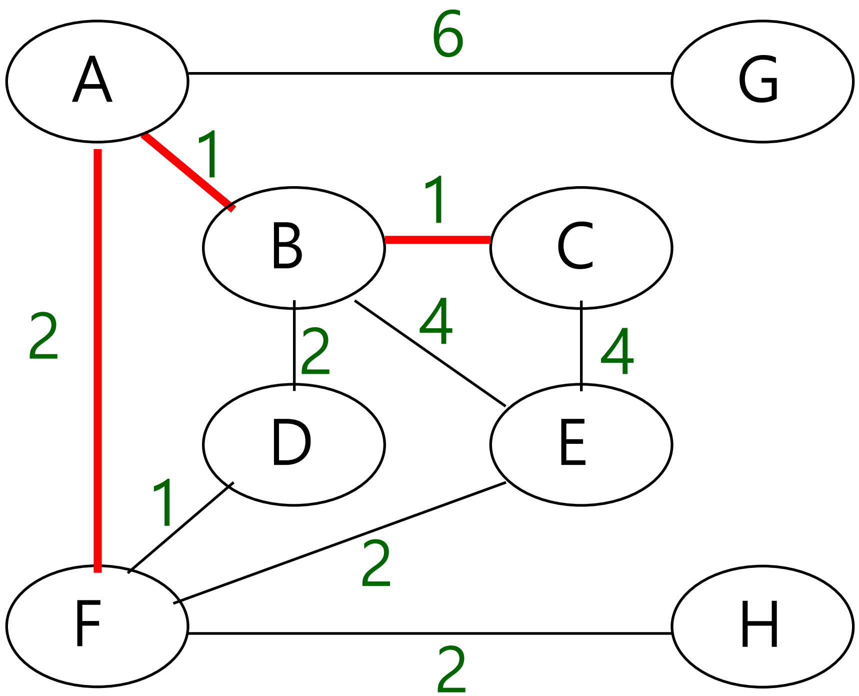
Prim's Algorithm: Illustrated (cont'd)



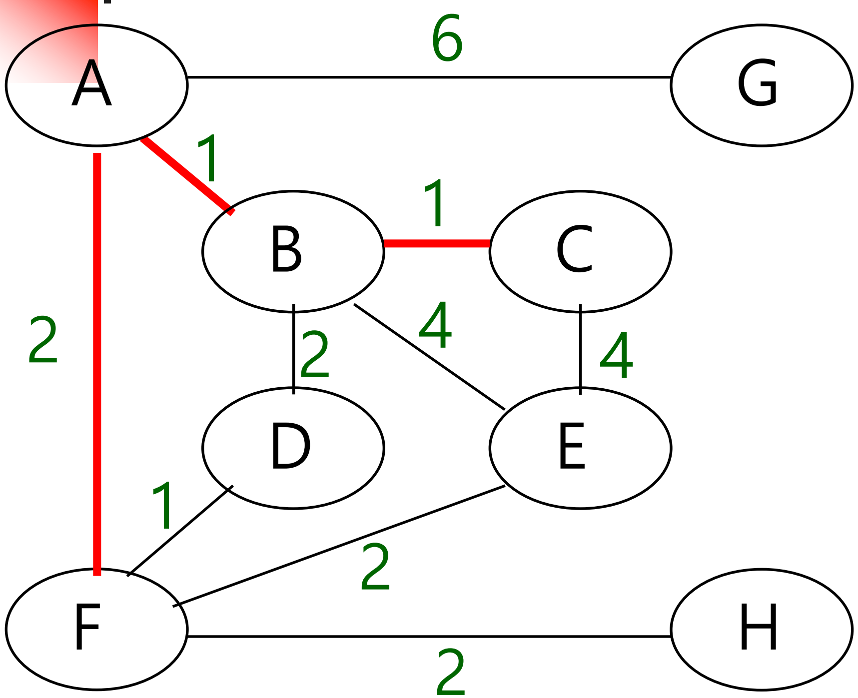


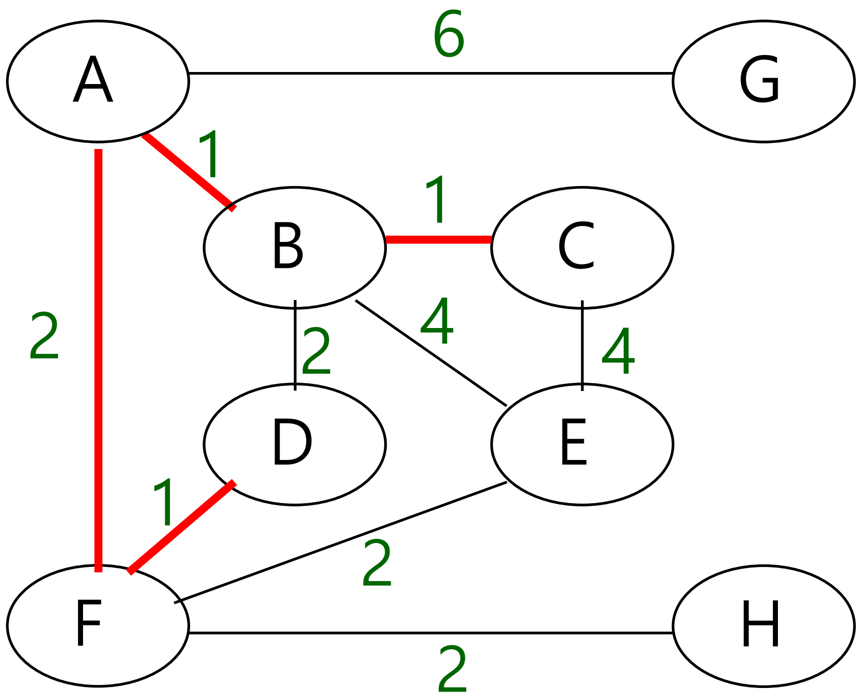
Prim's Algorithm: Illustrated (cont'd)



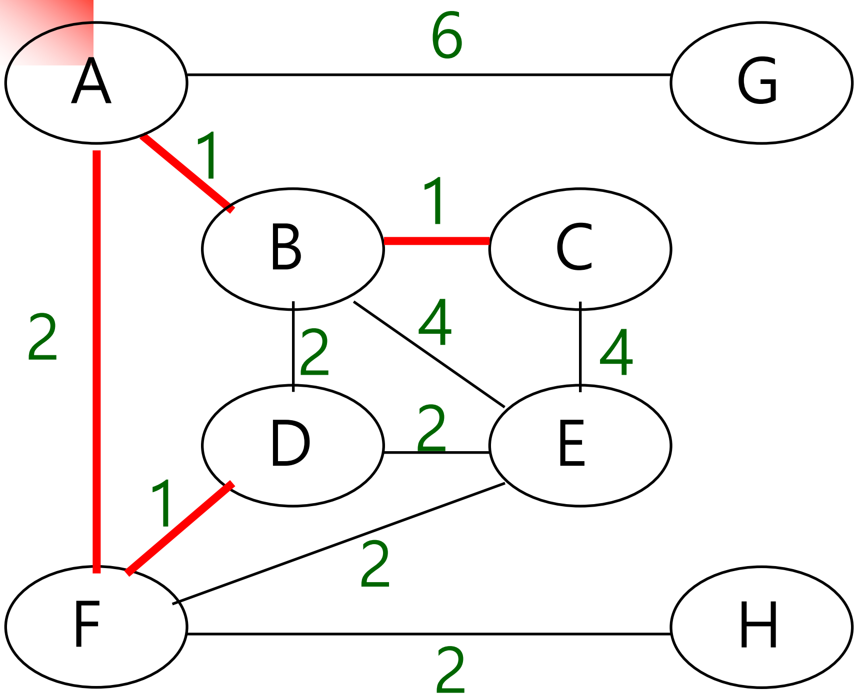


Prim's Algorithm: Illustrated (cont'd)

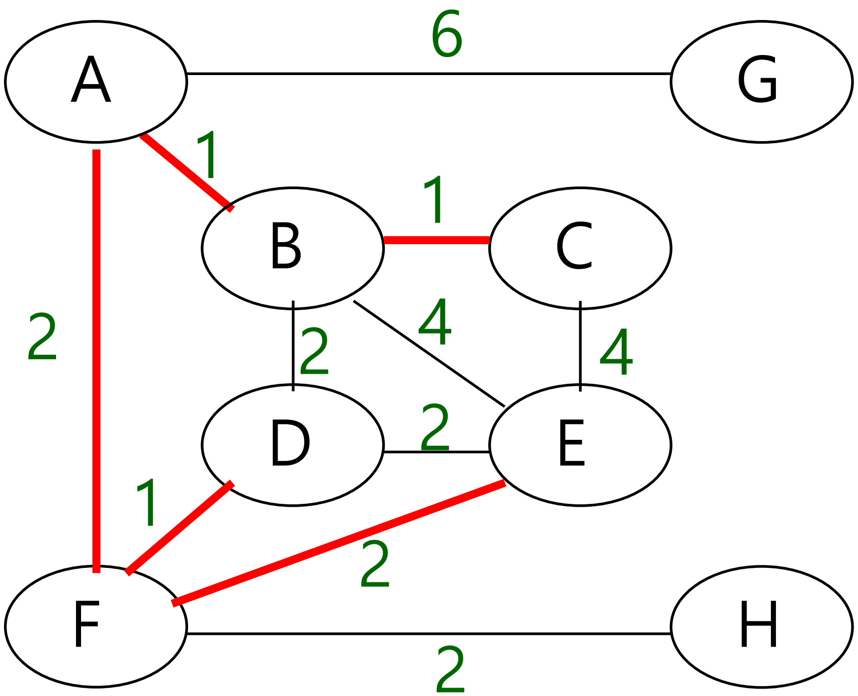




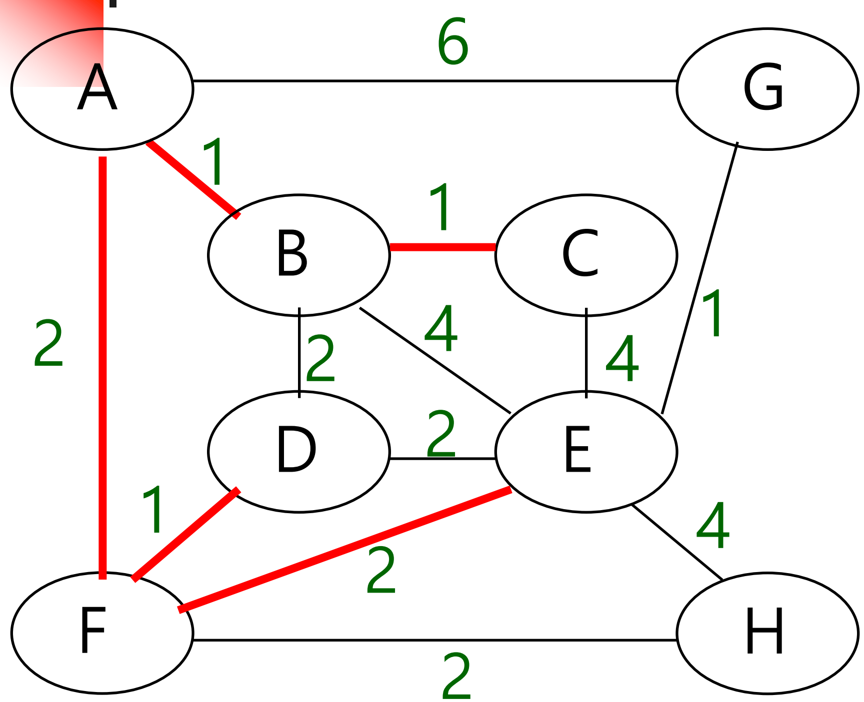
Prim's Algorithm: Illustrated (cont'd)

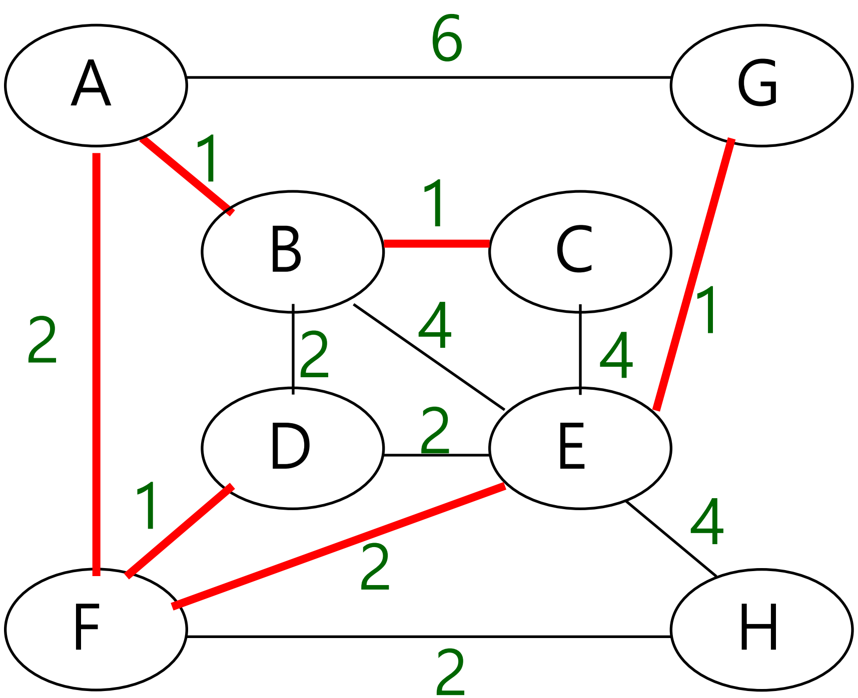


Do NOT pick
the B-D edge.

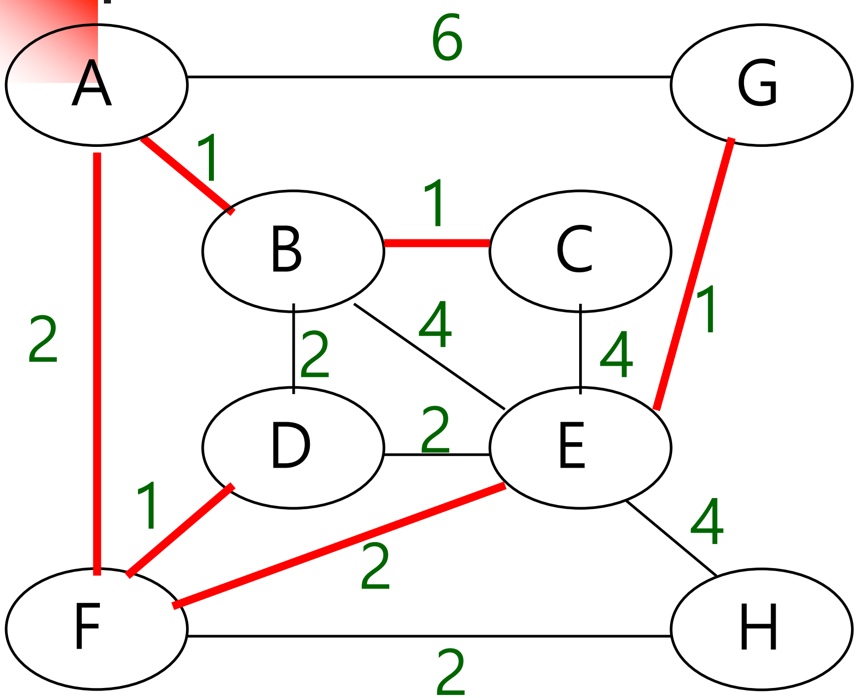


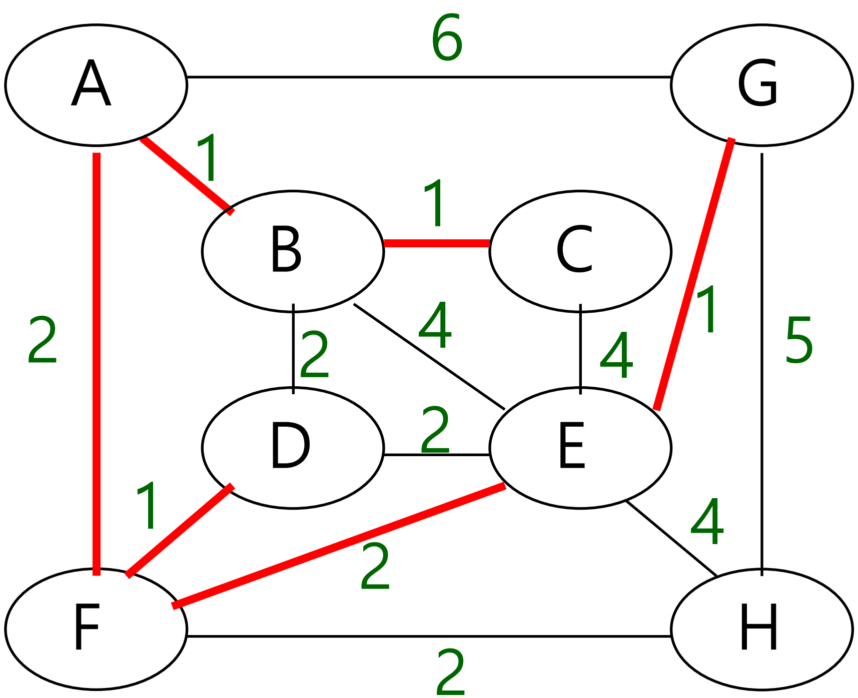
Prim's Algorithm: Illustrated (cont'd)



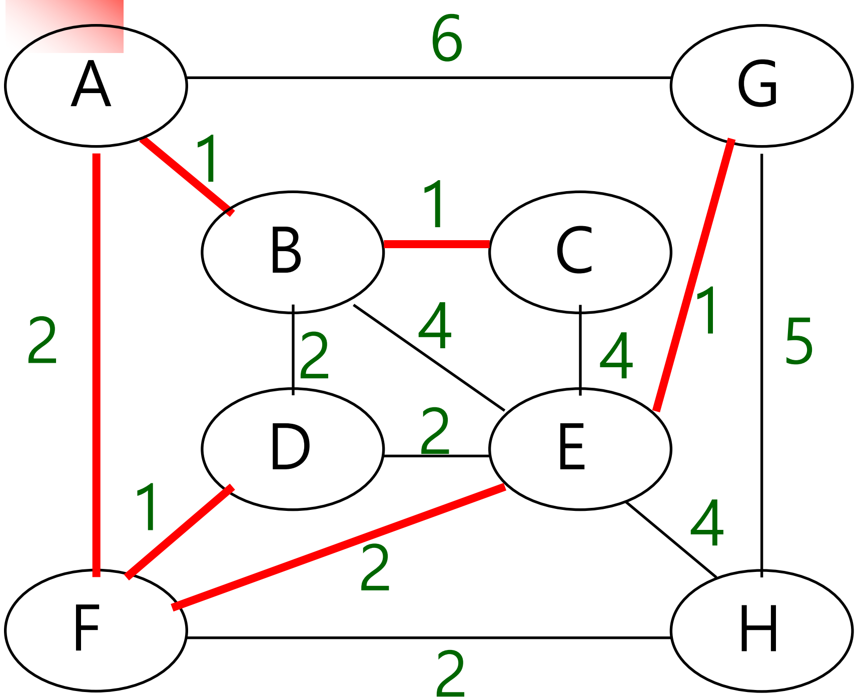


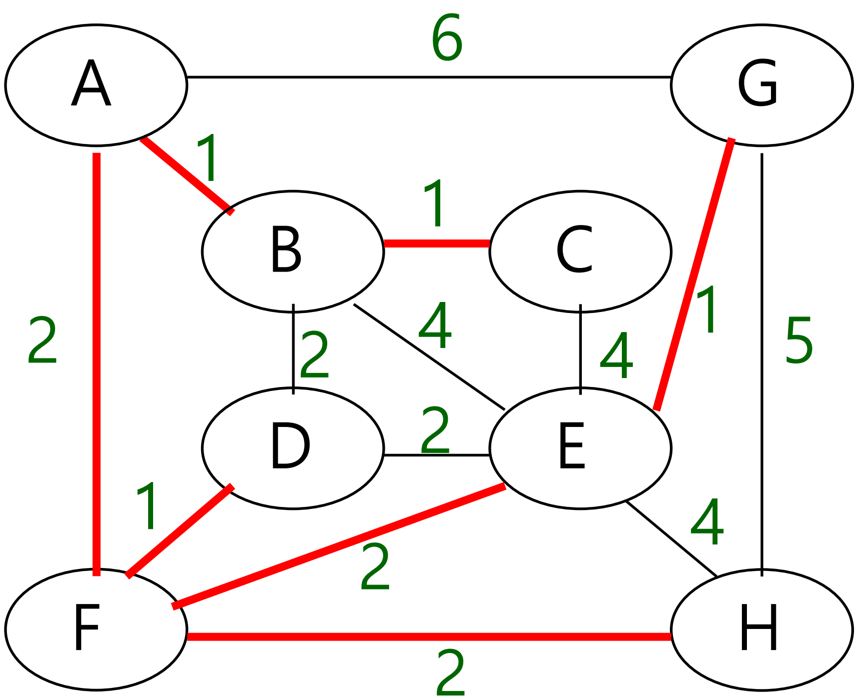
Prim's Algorithm: Illustrated (cont'd)





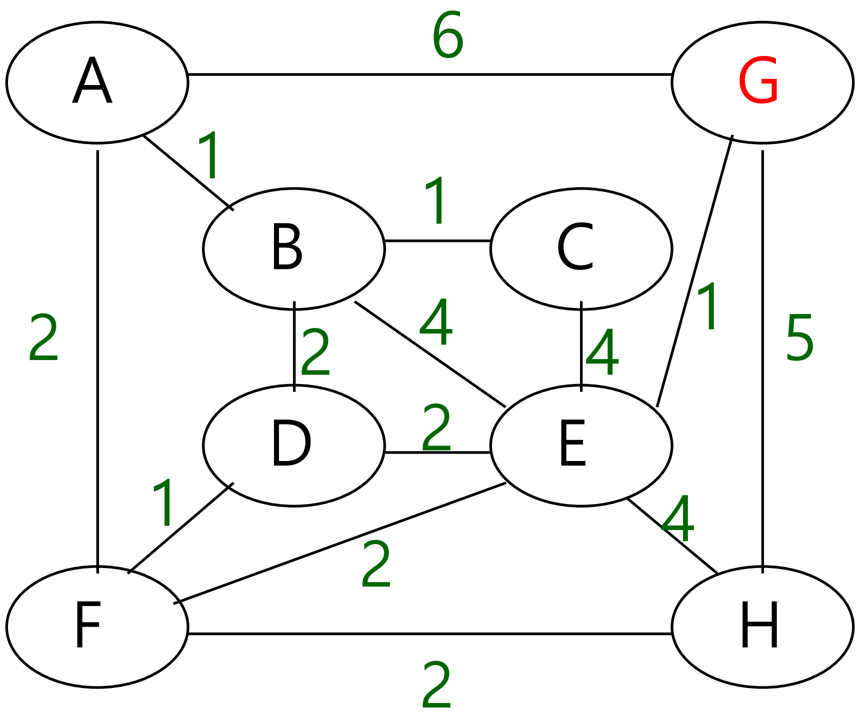
Prim's Algorithm: Illustrated (cont'd)



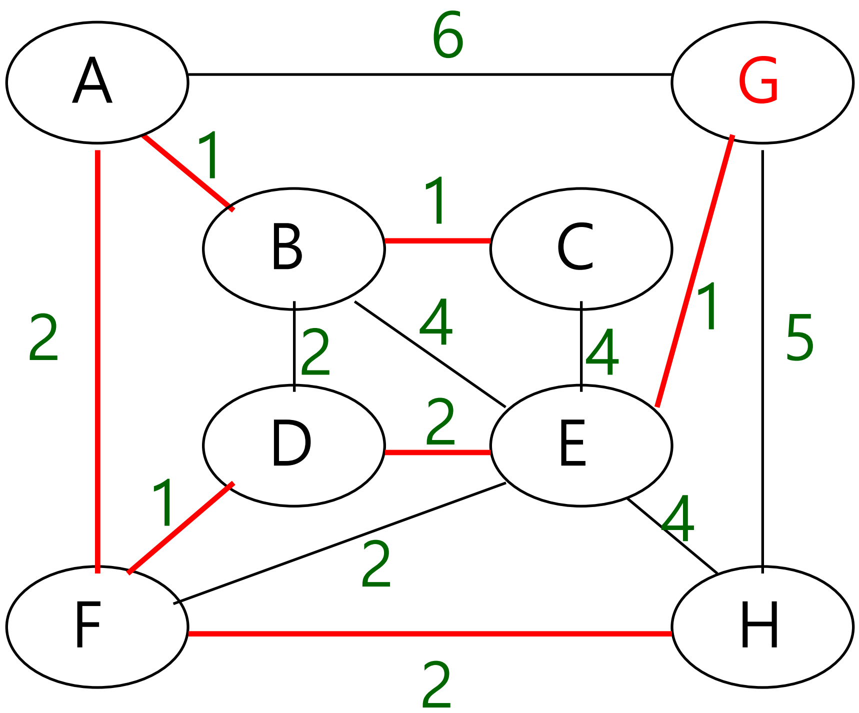


total = 10

Prim's Algorithm: Exercise 3



Prim's Algorithm:Solution

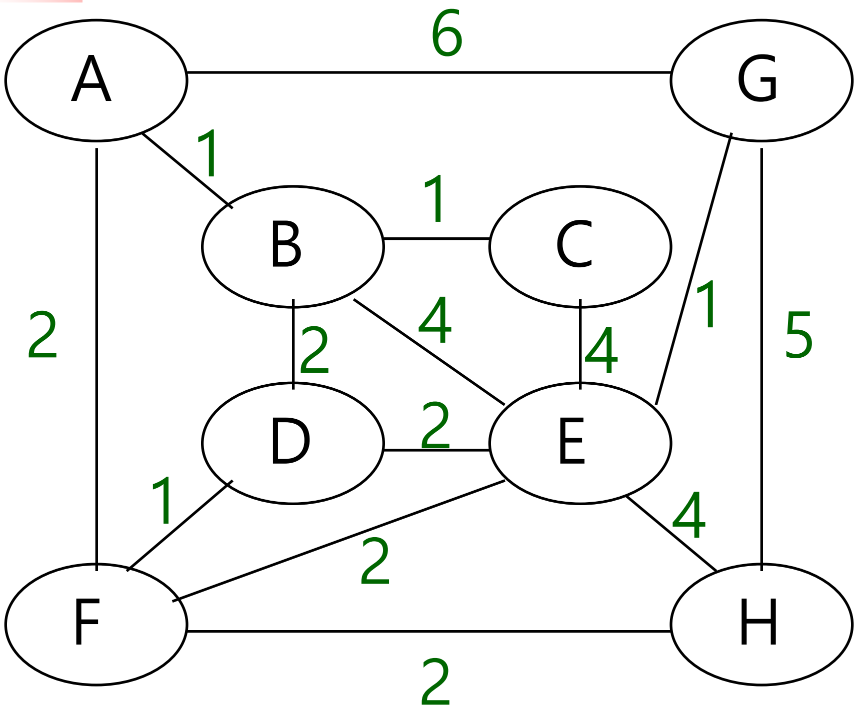


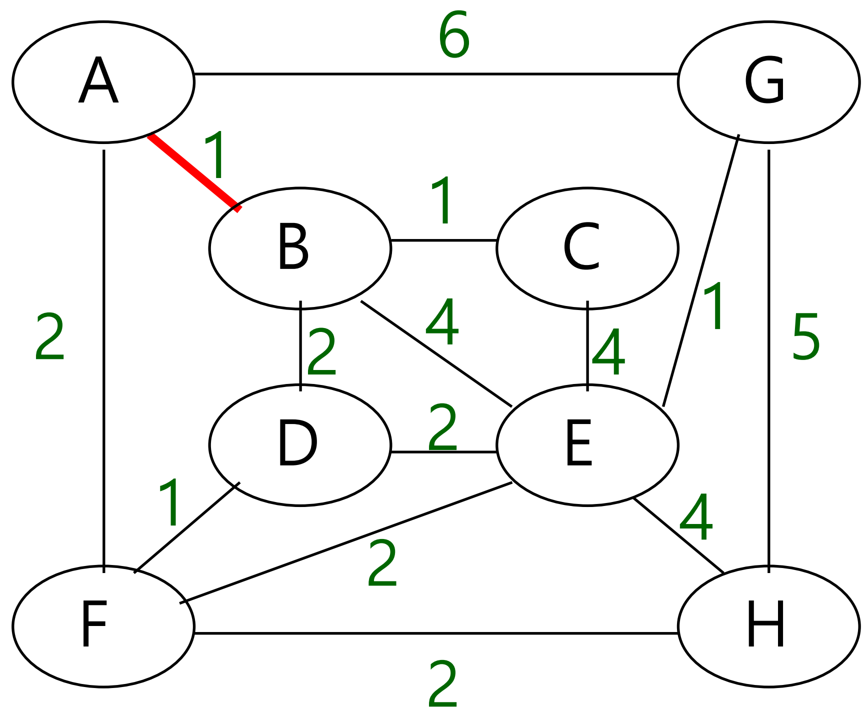
Kruskal's Algorithm

- Examine all edges of an undirected graph.
- Pick a minimum-weight edge, avoiding cycles.
  - (And insert the new node into a list of visited nodes)
- Continue until all nodes of the graph have been included (in the list of visited nodes).

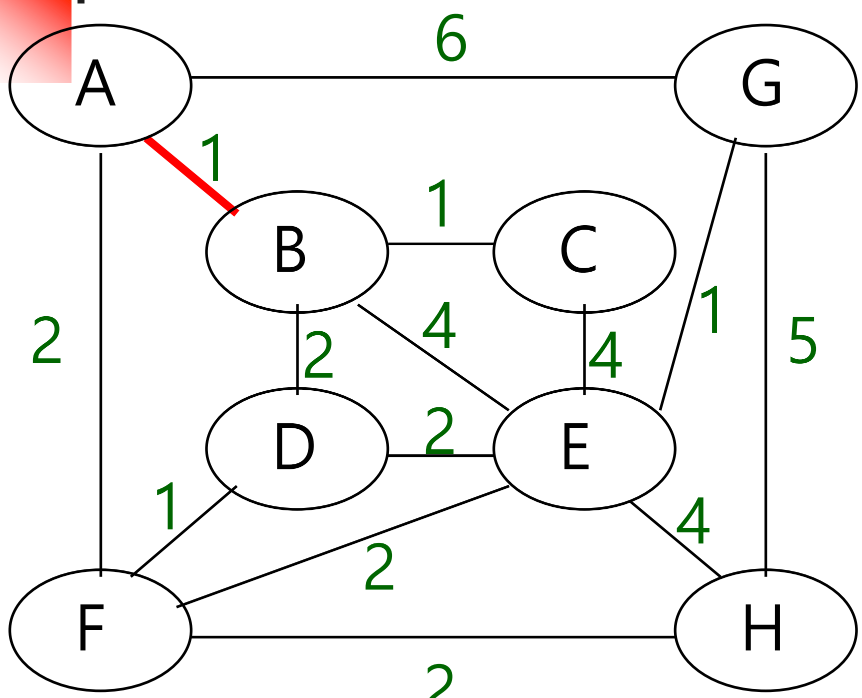
- Greedy algorithm
  - At each step, selects the minimum-weight edge among all edges

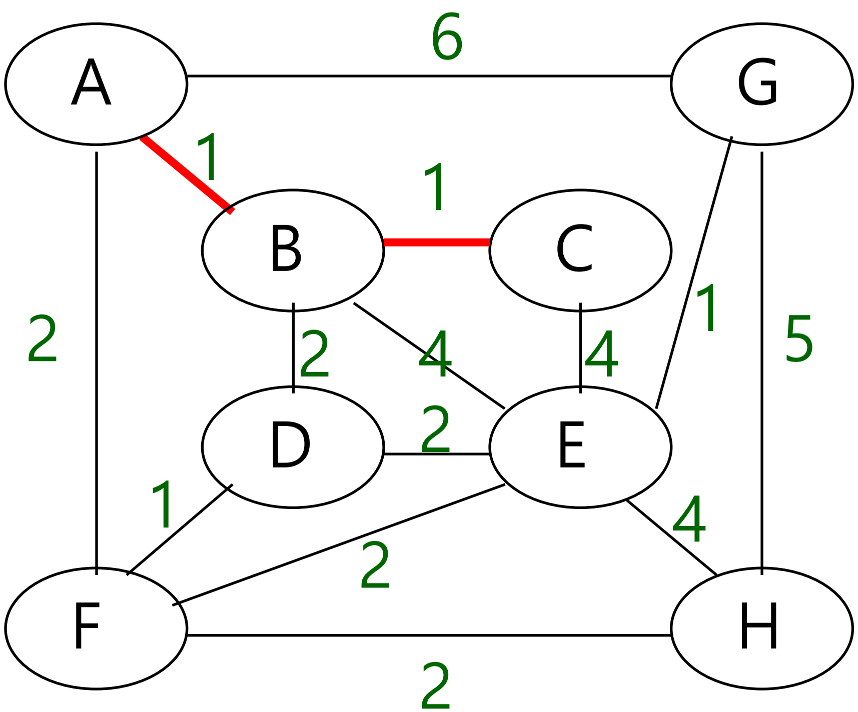
Kruskal's Algorithm: Illustrated



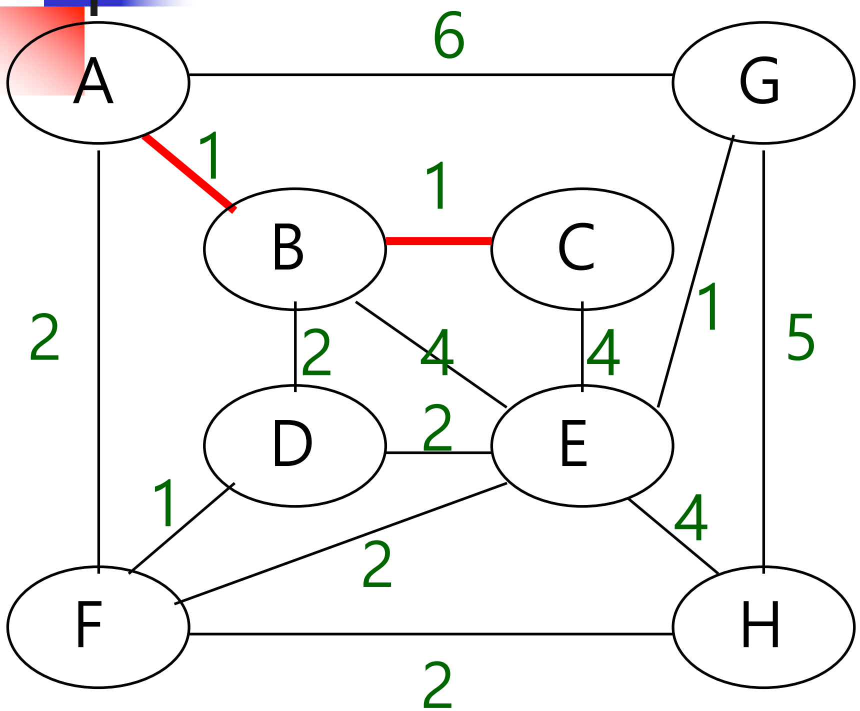


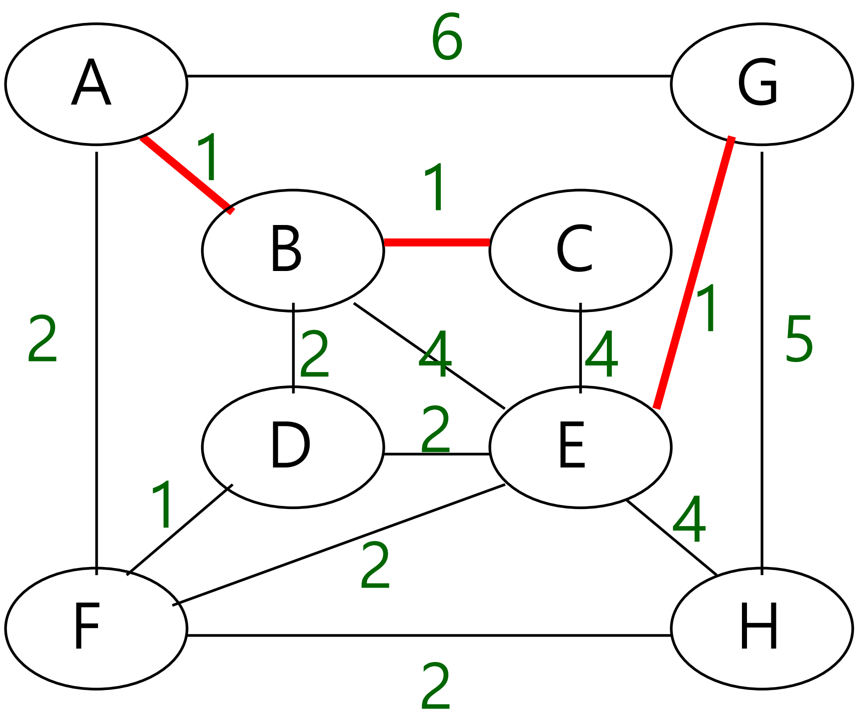
Kruskal's Algorithm: Illustrated (cont'd)



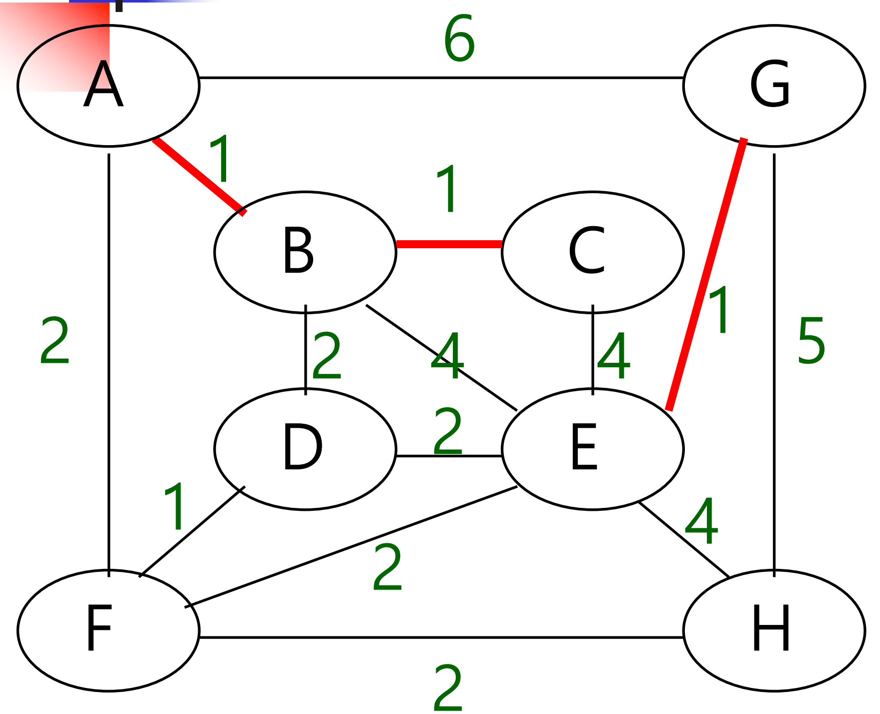


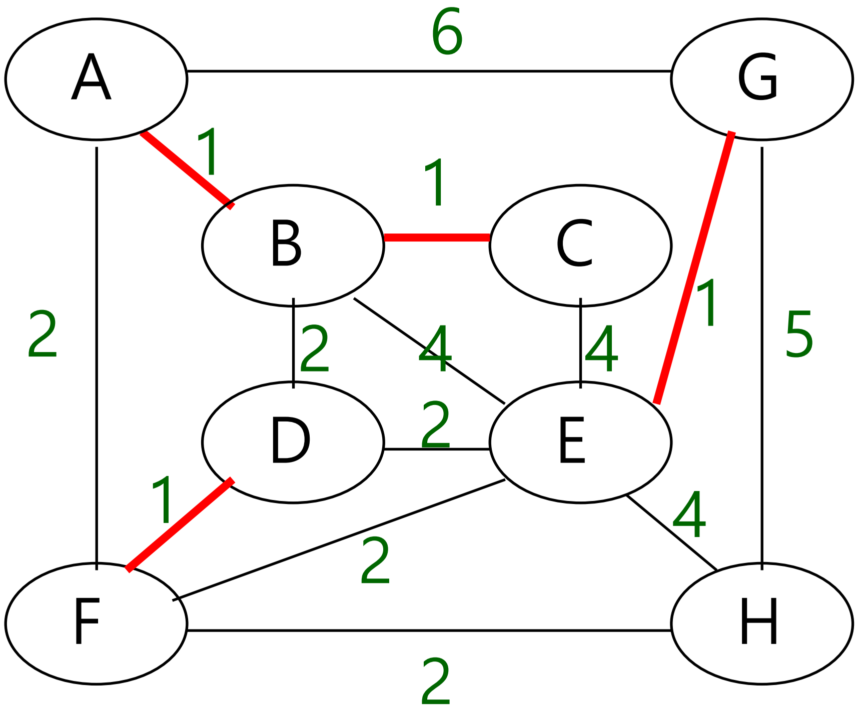
Kruskal's Algorithm: Illustrated (cont'd)



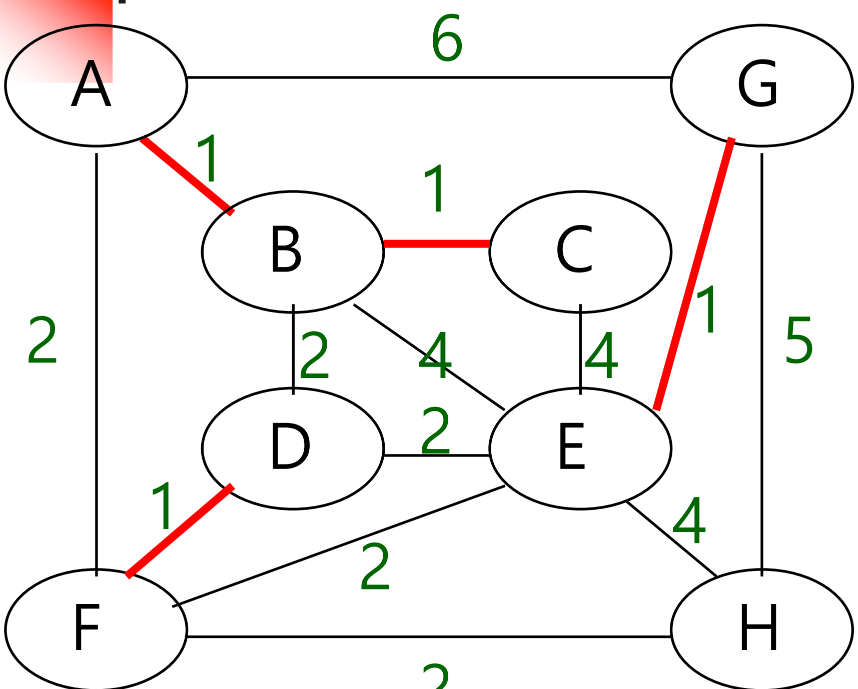


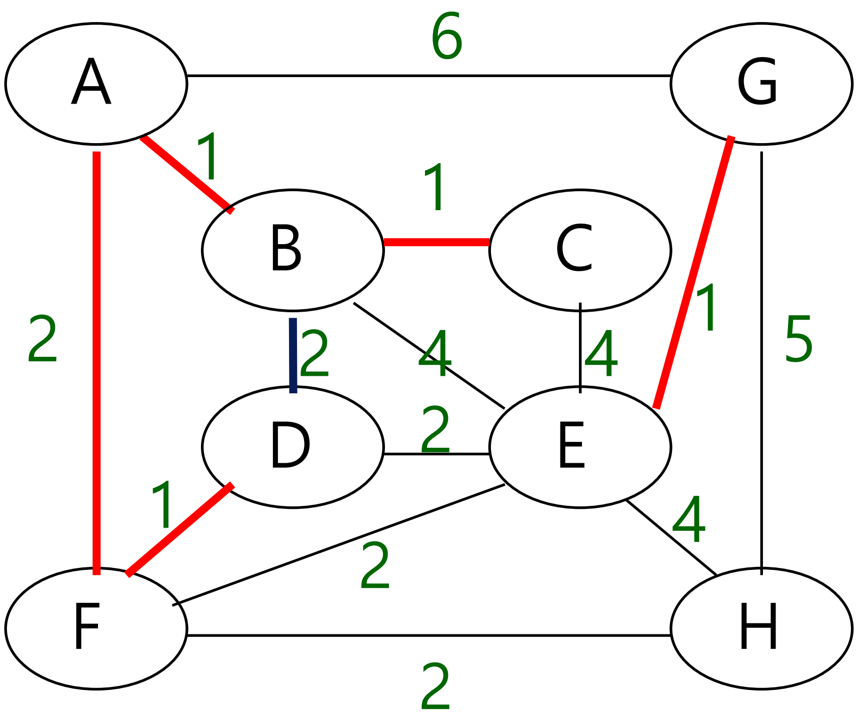
Kruskal's Algorithm: Illustrated (cont'd)





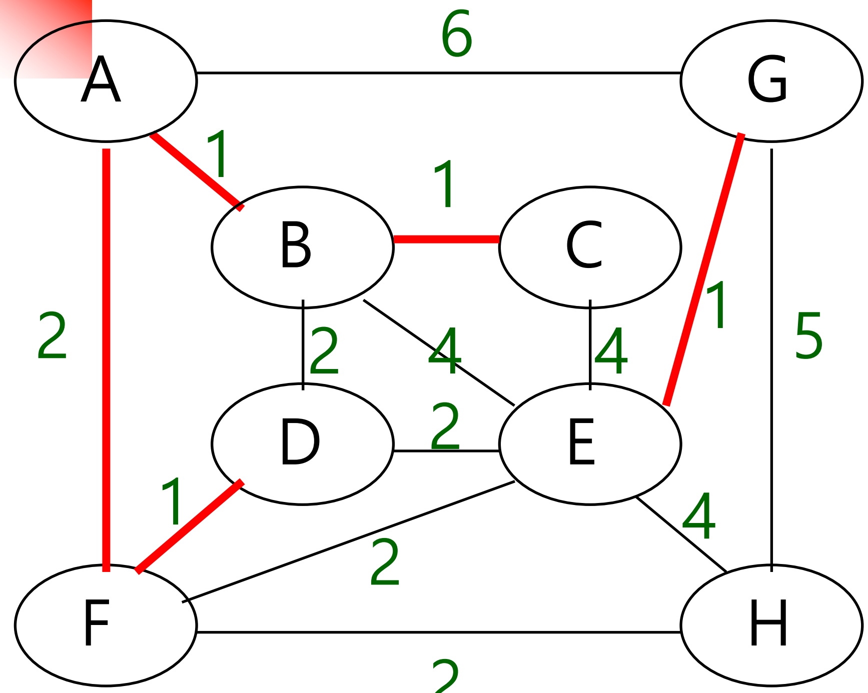
Kruskal's Algorithm: Illustrated (cont'd)

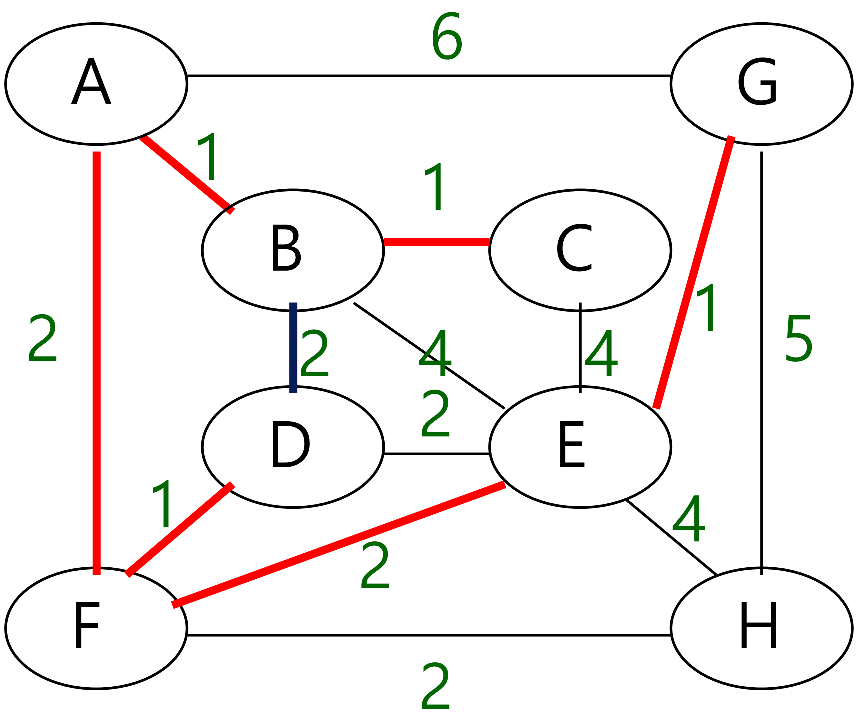




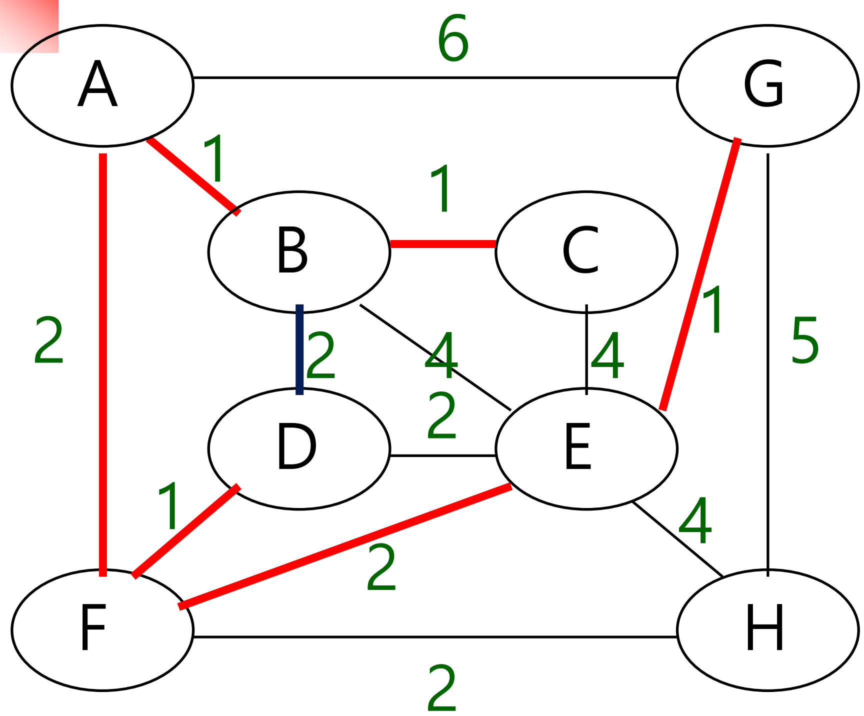
Do NOT pick
the B-D edge.

Kruskal's Algorithm: Illustrated (cont'd)

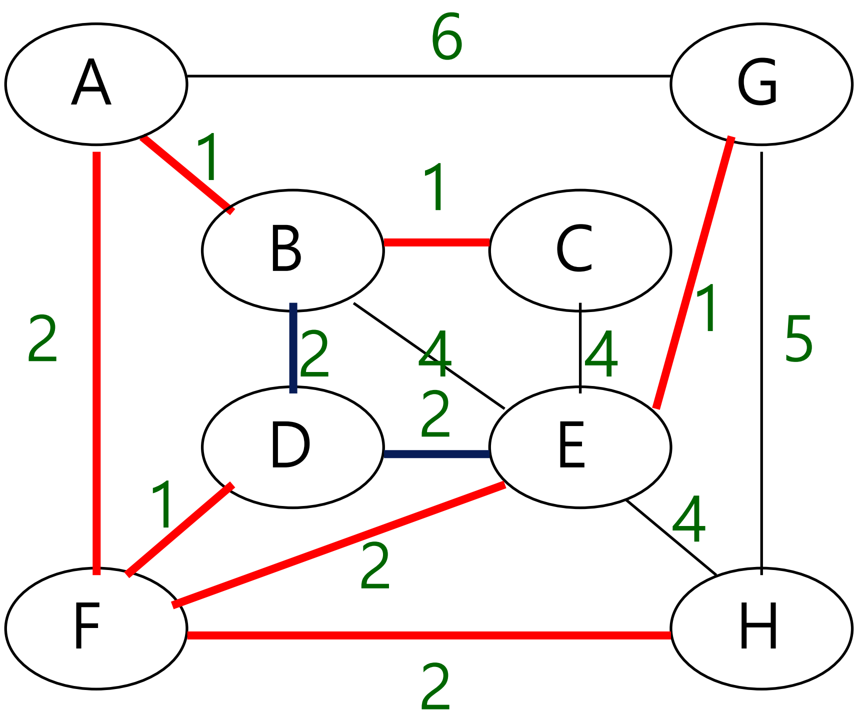




Kruskal's Algorithm: Illustrated (cont'd)

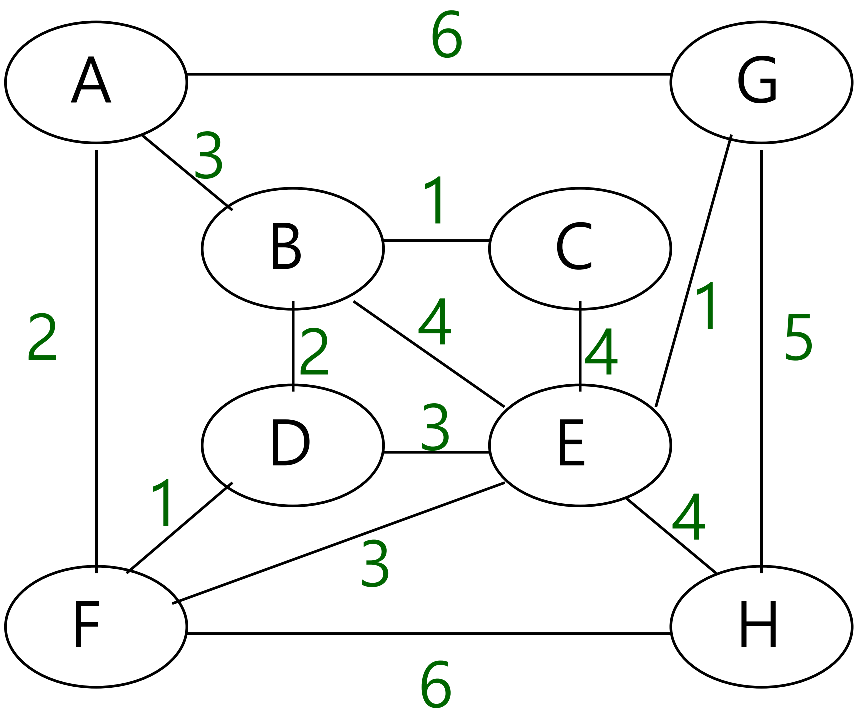


Do NOT pick the D-E edge.

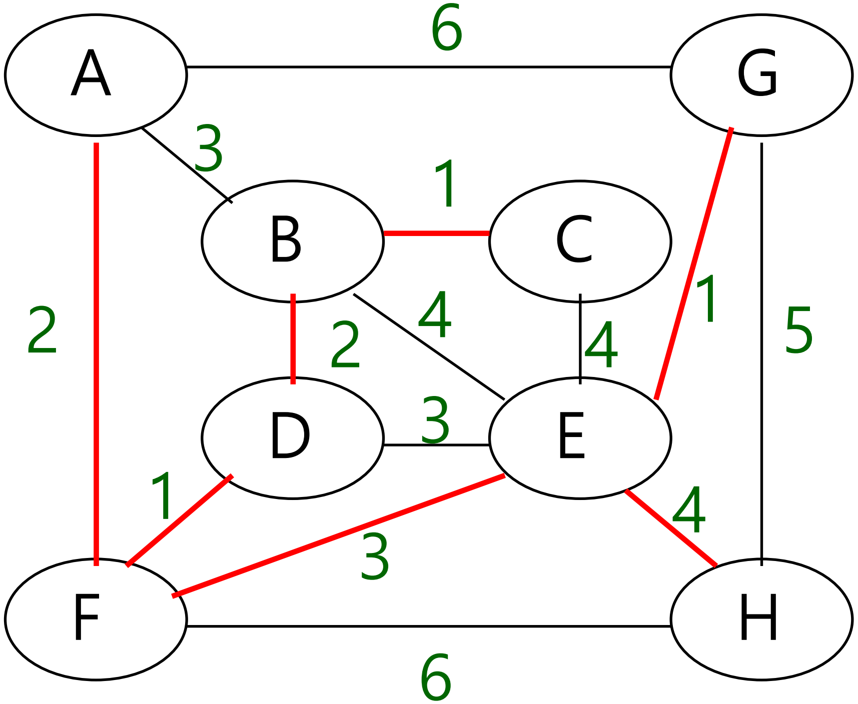


total = 10

Kruskal's Algorithm: Exercise 4

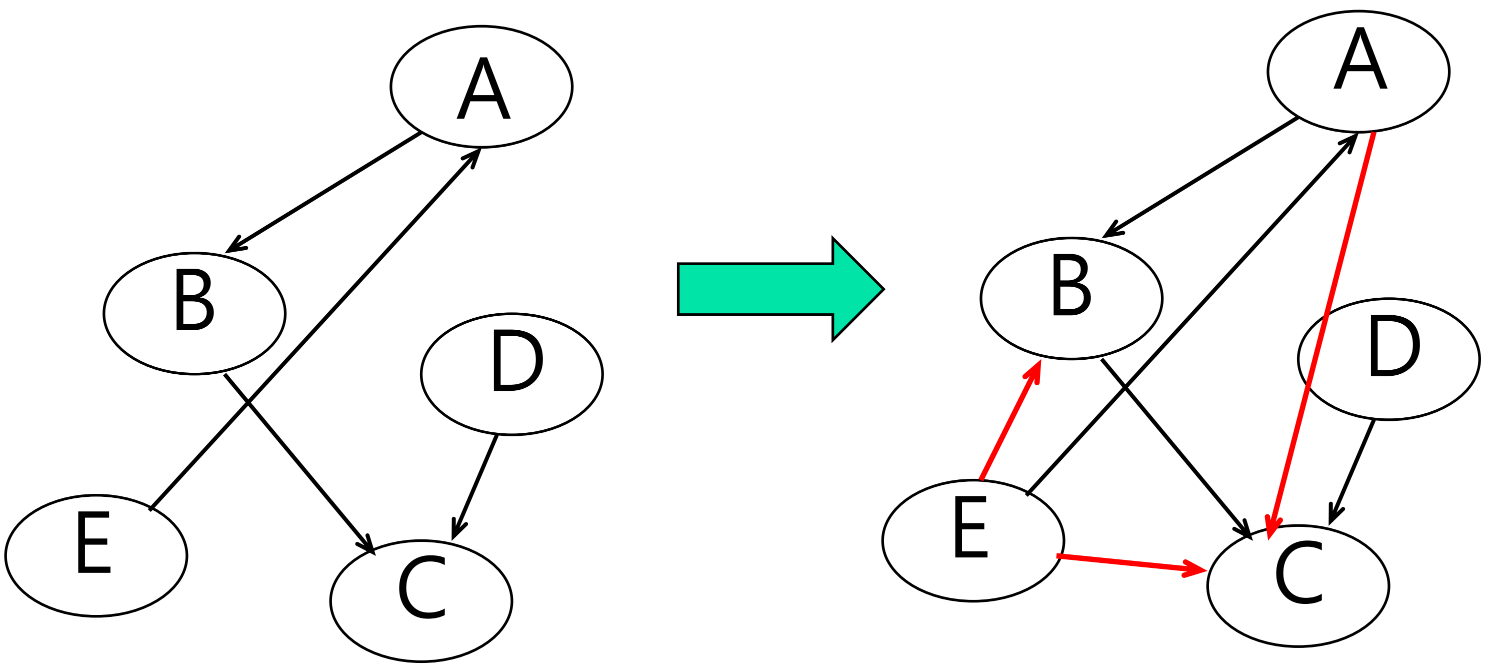


Kruskal's Algorithm: Solution



Transitive Closure Algorithms

- Determining direct and indirect paths between two nodes of a connected directed graph



transitive closure

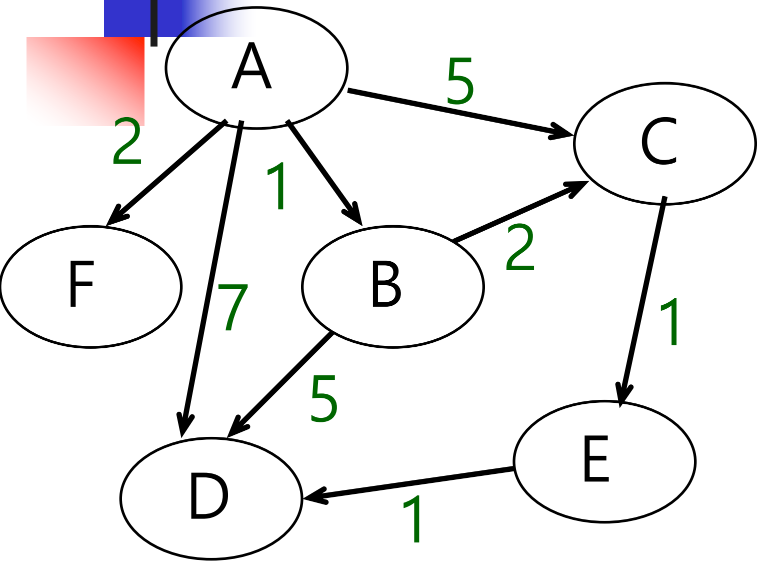
Popular Algorithms

- Dijkstra’s algorithm
- Warshall’s algorithm

Dijkstra’s Algorithm

- Determine a “least cost” path from node A and node B on a directed graph.
- A kind of minimum spanning tree algorithm, but does not span the entire graph and edges are directed.
- Performance: $O(N^2)$

Dijkstra's Algorithm: Illustrated

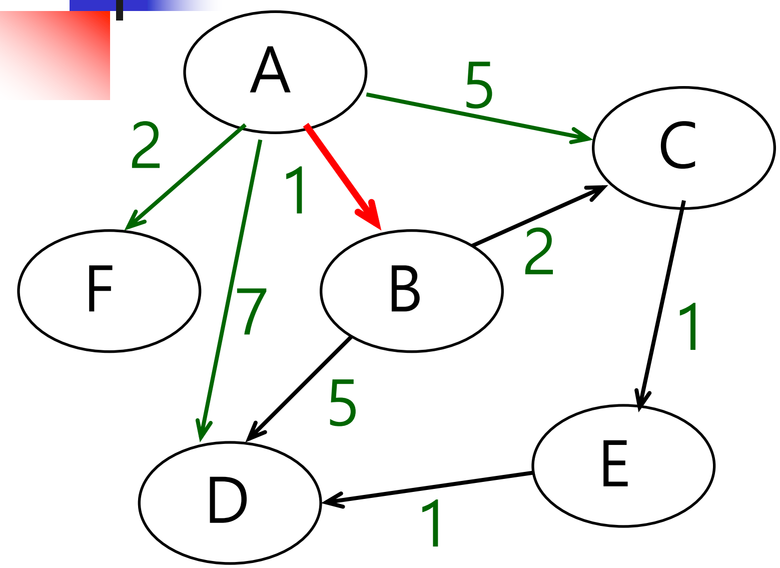


adjacency matrix

	A	B	C	D	E	F
A	x	1	5	7	-	2
B	-	x	2	5	-	-
C	-	-	x	-	1	-
D	-	-	-	x	-	-
E	-	-	-	1	x	-
F	-	-	-	-	-	x

Find the lowest cost path from A to D.

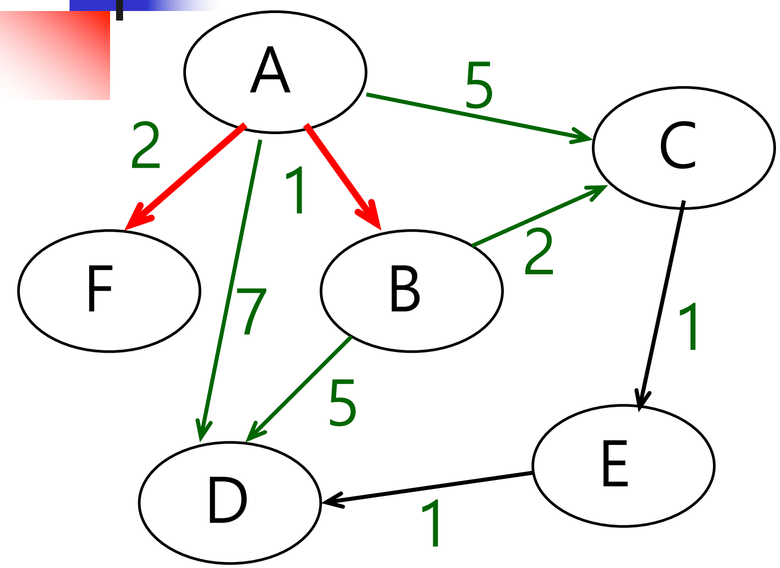
Dijkstra's Algorithm: Illustrated (cont'd)



path	B	C	D	E	F
A,B	1	5	7	-	2

	A	B	C	D	E	F
A	x	1	5	7	-	2
B	-	x	2	5	-	-
C	-	-	x	-	1	-
D	-	-	-	x	-	-
E	-	-	-	1	x	-
F	-	-	-	-	-	x

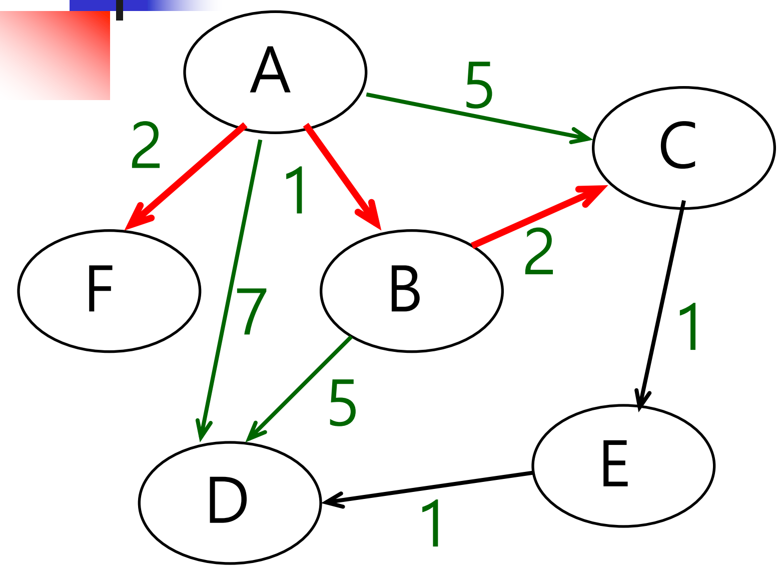
Dijkstra's Algorithm: Illustrated (cont'd)



path	B	C	D	E	F
A,B	1	5	7	-	2
A,F		3B	6B	-	2

	A	B	C	D	E	F
A	x	1	5	7	-	2
B	-	x	2	5	-	-
C	-	-	x	-	1	-
D	-	-	-	x	-	-
E	-	-	-	1	x	-
F	-	-	-	-	-	x

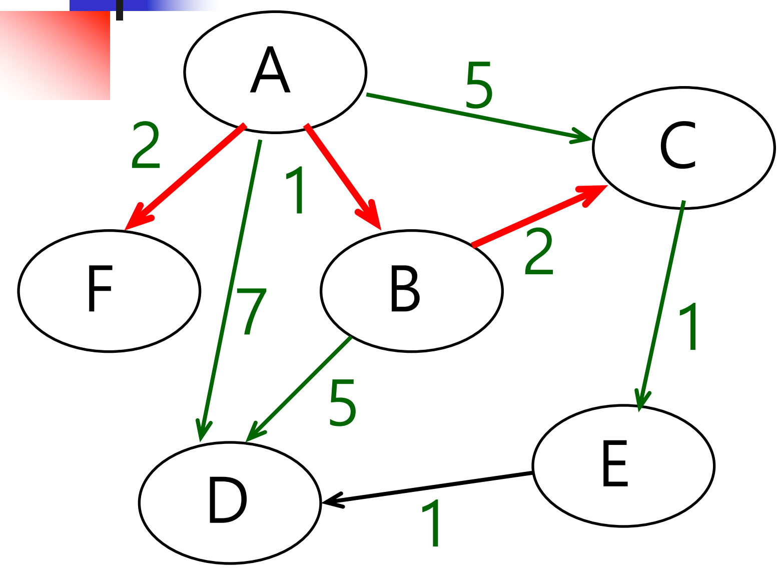
Dijkstra's Algorithm: Illustrated (cont'd)



	A	B	C	D	E	F
A	x	1	5	7	-	2
B	-	x	2	5	-	-
C	-	-	x	-	1	-
D	-	-	-	x	-	-
E	-	-	-	1	x	-
F	-	-	-	-	-	x

path	B	C	D	E	F
A,B	1	5	7	-	2
A,F		3B	6B	-	2
A,B,C		3B	6B	-	

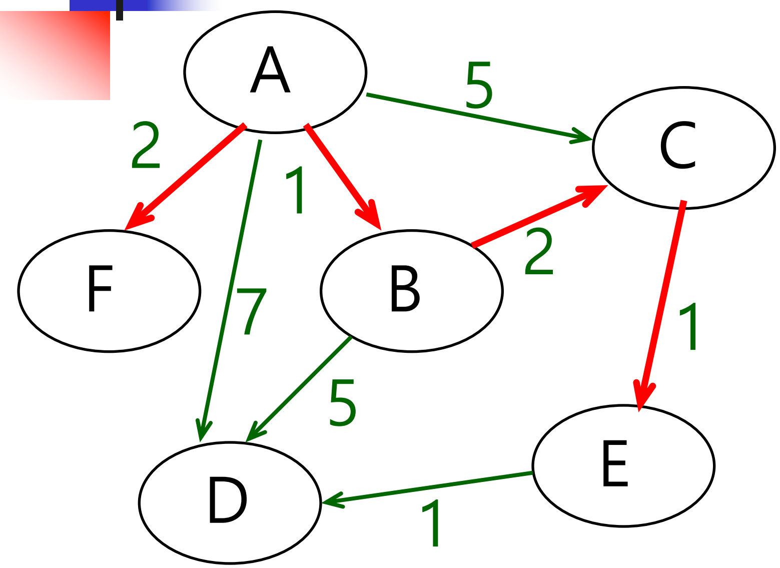
Dijkstra's Algorithm: Illustrated (cont'd)



	A	B	C	D	E	F
A	x	1	5	7	-	2
B	-	x	2	5	-	-
C	-	-	x	-	1	-
D	-	-	-	x	-	-
E	-	-	-	1	x	-
F	-	-	-	-	-	x

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

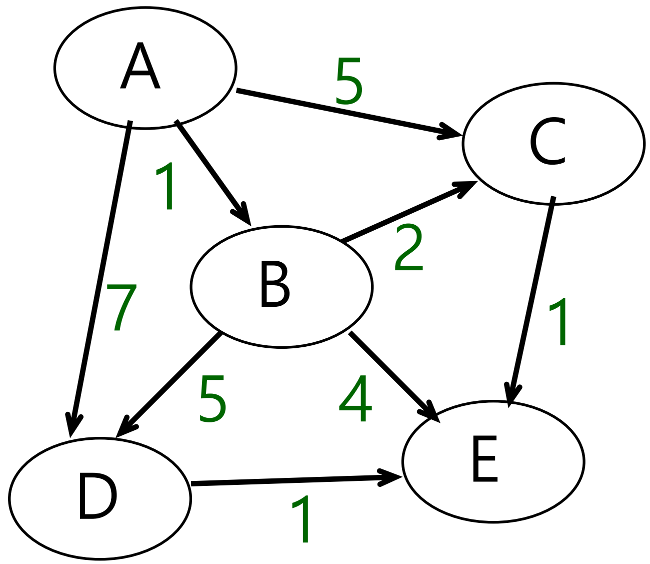
Dijkstra's Algorithm: Illustrated (cont'd)



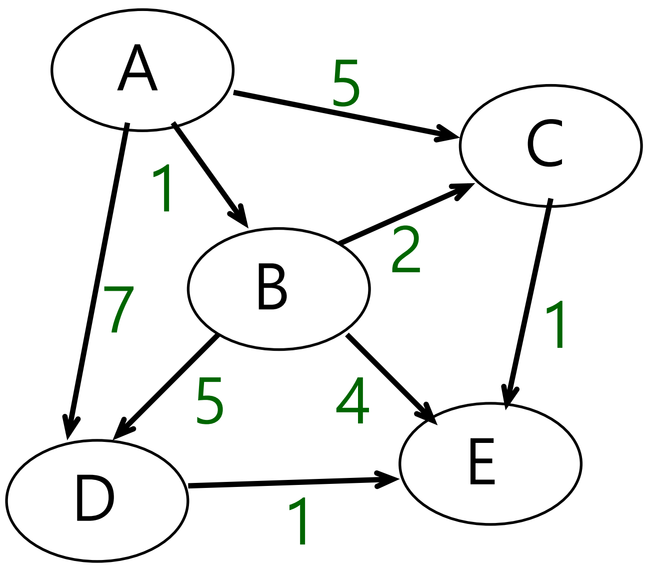
path	B	C	D	E	F
A,B	1	5	7	-	2
A,F		3B	6B	-	2
A,B,C		3B	6B	-	
A,B,C,E			6B	4C	
A,B,C,E,D			5E		

lowest cost path
A-B-C-E-D
cost = 5

Dijkstra's Algorithm: Exercise 5 (Lowest Cost Path from A to E)



Dijkstra's Algorithm:Solution (Lowest Cost Path from A to E)



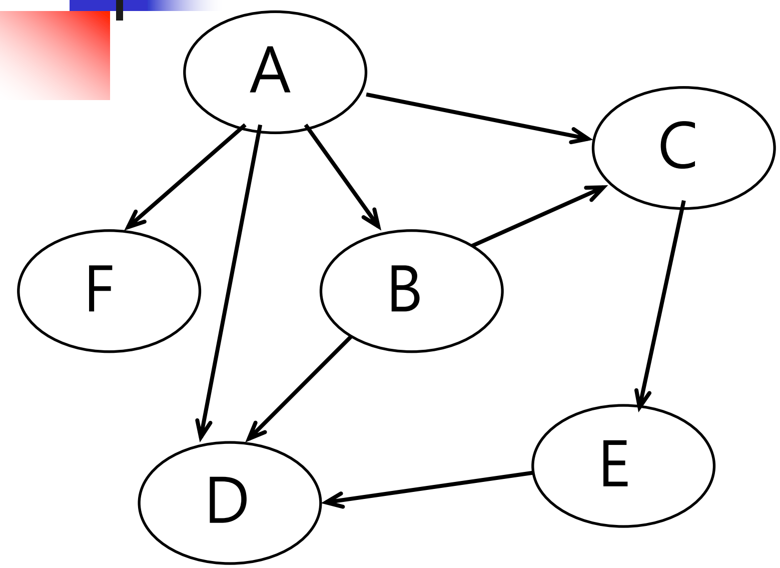
path	B	C	D	E
A,B	1	5	7	-
A,B,C		3B	6B	5B
A,B,C,E			6B	4C

lowest cost path
A-B-C-E
cost = 4

Warshall's Algorithm

- Computes transitive closure using an adjacency matrix
- Performance: $O(N^3)$

Warshall's Algorithm: Illustrated



adjacency matrix

	A	B	C	D	E	F
A	0	1	1	1	0	1
B	0	0	1	1	0	0
C	0	0	0	0	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	0
F	0	0	0	0	0	0

Scan the matrix
from left to right
from top to bottom.

Warshall's Algorithm: Illustrated (cont'd)

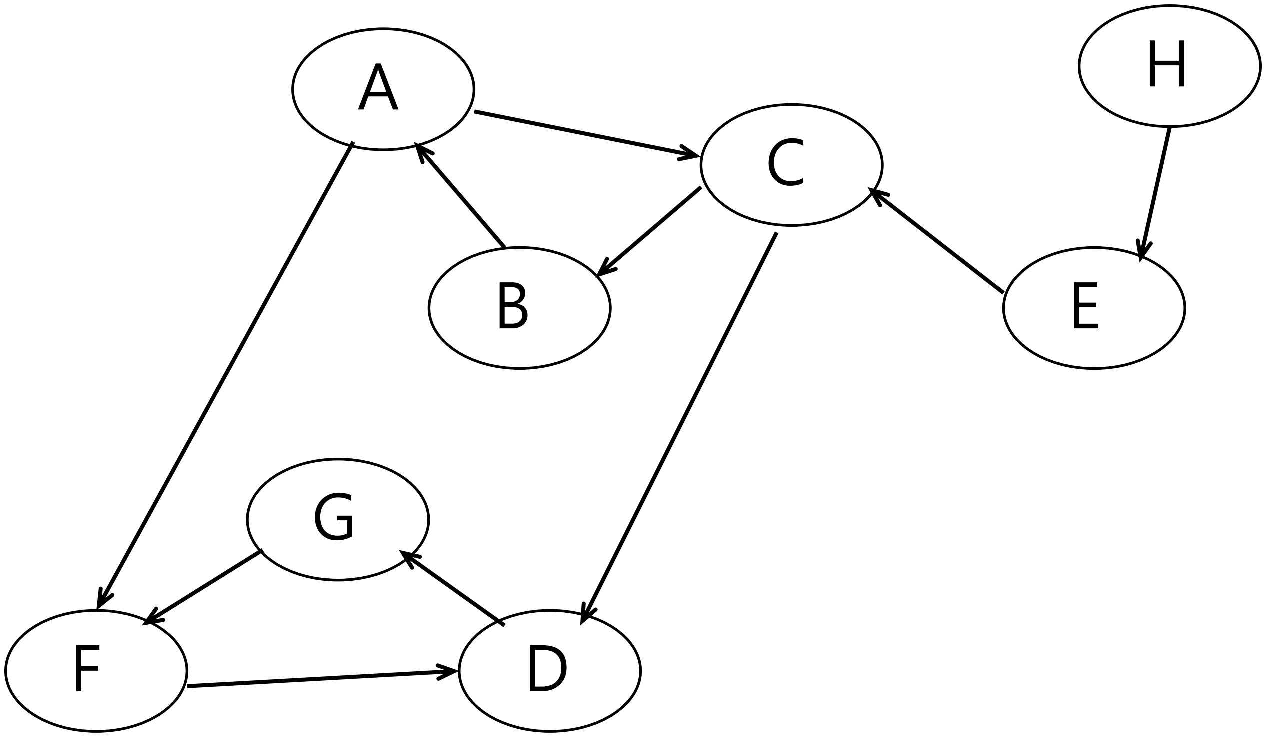
A->C=1
C->E=1
means
A->E=1

B->C=1
C->E=1
means
B->E=1

C->E=1
E->D=1
means
C->D=1

	A	B	C	D	E	F
A	0	1	1	1	1	1
B	0	0	1	1	1	0
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	0
F	0	0	0	0	0	0

Example: Find a Transitive Closure Using Warshall's Algorithm



Warshall's Algorithm: Illustrated (cont'd)

	A	B	C	D	E	F	G	H
A	0	0	1	0	0	1	0	0
B	1	0	0	0	0	0	0	0
C	0	1	0	1	0	0	0	0
D	0	0	0	0	0	0	1	0
E	0	0	1	0	0	0	0	0
F	0	0	0	1	0	0	0	0
G	0	0	0	0	0	1	0	0
H	0	0	0	0	1	0	0	0

Warshall's Algorithm: Illustrated (cont'd)

B->A=1
means
B->C=1
B->F=1

C->B=1
means
C->A=1
C->F=1

	A	B	C	D	E	F	G	H
A	0	0	1	0	0	1	0	0
B	1	0	1	0	0	1	0	0
C	1	1	0	1	0	1	0	0
D	0	0	0	0	0	0	1	0
E	0	0	1	0	0	0	0	0
F	0	0	0	1	0	0	0	0
G	0	0	0	0	0	1	0	0
H	0	0	0	0	1	0	0	0

Warshall's Algorithm: Illustrated (cont'd)

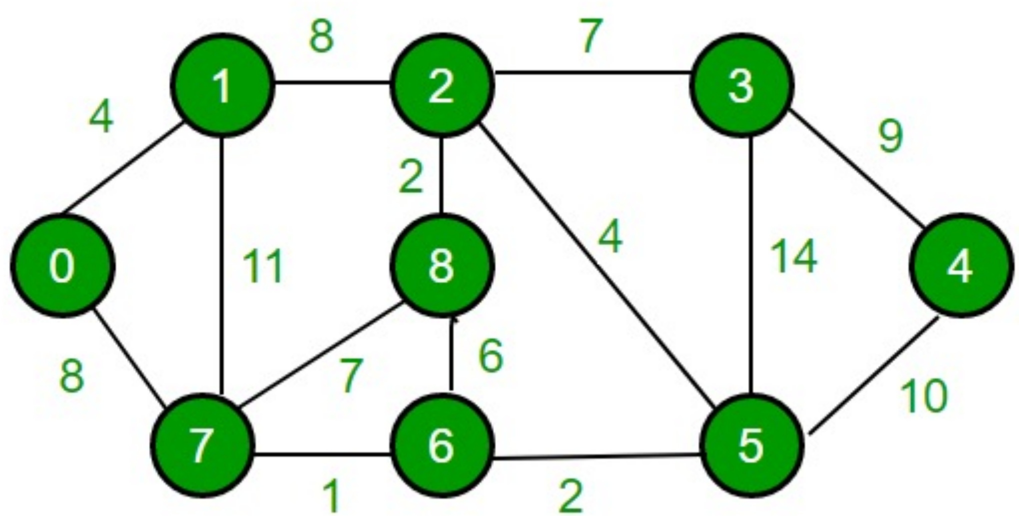
End

	A	B	C	D	E	F	G	H
A	0	1	1	1	0	1	1	0
B	1	0	1	1	0	1	1	0
C	1	1	0	1	0	1	1	0
D	0	0	0	0	0	1	1	0
E	1	1	1	1	0	1	1	0
F	0	0	0	1	0	0	1	0
G	0	0	0	1	0	1	0	0
H	1	1	1	1	1	1	1	0

Written Homework 7

- Find the shortest path and its length from Node 0 to ALL THE OTHER NODES in the graph using Dijksta's algorithm.
  i.e., Find the solution for each of the following:
  Node 0 to Node 1: shortest path and its length
  Node 0 to Node 2: shortest path and its length

  ……..
  Node 0 to Node 8 : : shortest path and its length



End of Lecture