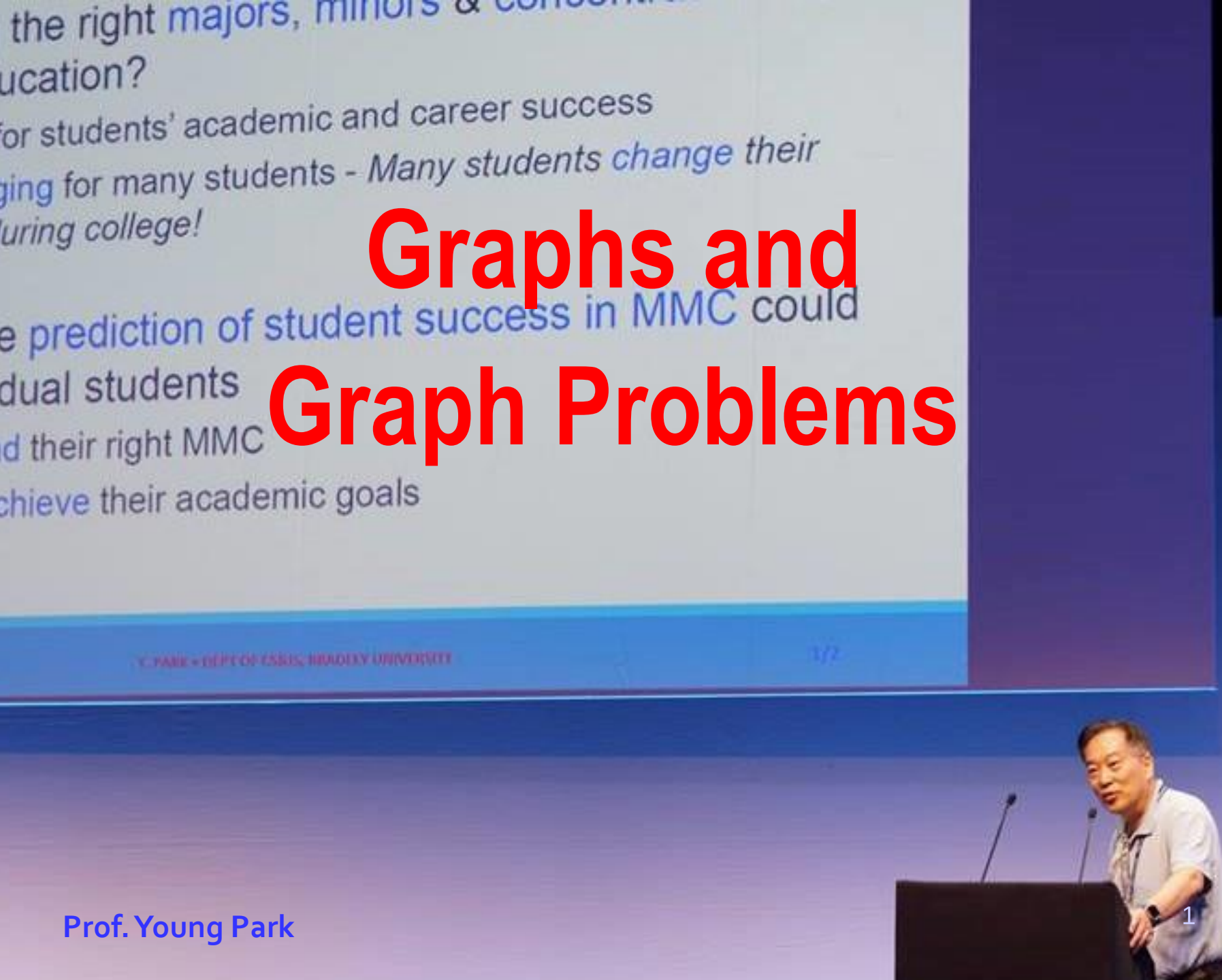
Graphs and Graph Problems



Prof. Young Park

Graphs as Non-linear Data Structures

What Is a Graph?

- A nonlinear data structure consisting of a set V of nodes (vertices) and a set E of links between the nodes (edges or arcs).
  - G = (V, E)
  - V = {v1, v2, …, vn}
  - E = { (vi,vj) }
- An edge can connect any pair of vertices.
  - v is **adjacent** to u when **(u, v)** in E.
- **Like lists & trees, but more general!**

What Graphs?

- Undirected graphs
- Directed graphs
- Acyclic graphs
- Connected graphs
- Labeled (Weighted) graphs
- Sparse/Dense graphs
- ...

Paths in Graphs

- A path in a graph is
  - An alternating sequence of vertices and edges in which all edges and vertices are distinct.
  - A sequence of vertices in which all vertices are distinct.
    - P = < v1, v2, … vk> where (vi,vj) in E
- The length of a path is
  - The number of edges on the path.

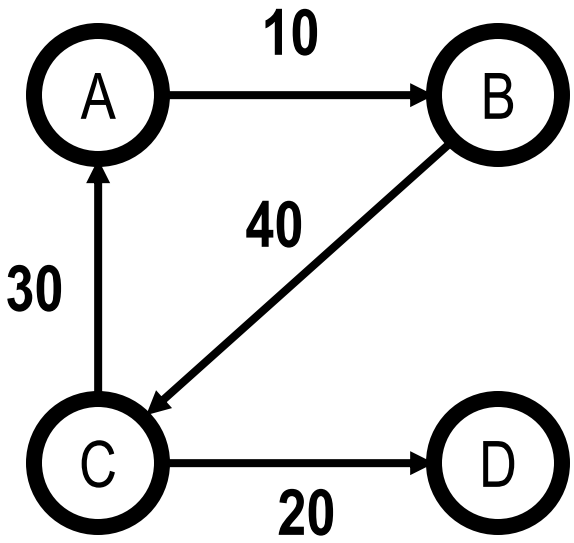
Cycles in Graphs

- A cycle in a graph is
  - ➔ A path in which the first vertex and the last vertex is the same.

Labeled Graphs

- A graph G = (V, E) is called a labeled graph
  - ➔ If its edges are assigned data of one kind of another.
    - ☞ labeled edges
  - ➔ If its vertices are assigned data of one kind of another.
    - ☞ labeled vertices
  - ➔ If its edges and vertices are assigned data of one kind of another.
    - ☞ labeled edges and vertices
  - ➔ The assigned some numerical value is called the weight or cost. (called **Weighted Graphs**)

Example: Edge-Weighted Graphs



Sparse Graphs vs Dense Graphs

- A graph G = (V, E) is said to be **sparse**
  - ➔If it has relatively few edges.
  - ➔$|E| = O(|V|)$ edges
- A graph G = (V, E) is said to be **dense**
  - ➔If it has many edges.
  - ➔$|E| = O(|V|^2)$ edges

Why Graphs in Problem Solving?

- Many real-world problems can be reduced to problems on graphs.
  - ➔Often a problem can be represented as a graph and the solution to the problem is obtained by solving a problem on the corresponding graph!
  - ➔Many problems can be solved by asking an appropriate question about paths in a graph!

Graphs vs Other Data Structures

- A graph is a generalized data structure of other data structures
  - ➔Lists
  - ➔Trees
  - ➔Heaps
  - ➔…

►QUIZ?

- Graphs vs. Trees?
- Dense graphs vs. Sparse graphs?

Representation of Graphs

How to Represent Graphs?

- An array-based representation
  - ➔Use an **adjacency matrix**
- A pointer-based representation
  - ➔Use an **adjacency list**

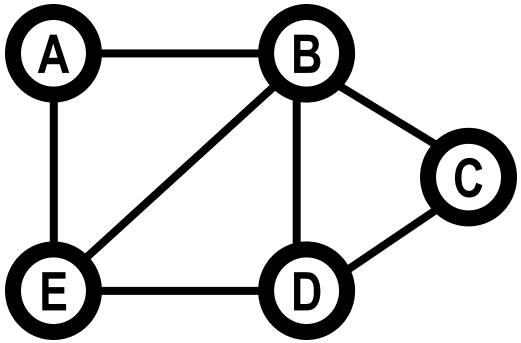
1. An Adjacency Matrix-based Representation

- Use a two-dimensional array - matrix
- For an undirected graph G = (V, E)
  - A[i][j] =
    - 1 if (vi,vj) in E
    - 0 otherwise
- For an edge-weighted undirected graph G = (V, E)
  - A[i][j] =
    - weight (numeric label) if (vi,vj) in E
    - ∞ otherwise
- A[i][j] = A[j][i]

An Adjacency Matrix-based Representation

- For a directed graph G = (V, E)
  - A[i][j] =
    - 1 if (vi,vj) in E
    - 0 otherwise
- For an edge-weighted directed graph G = (V, E)
  - A[i][j] =
    - weight (numeric label) if (vi,vj) in E
    - ∞ otherwise

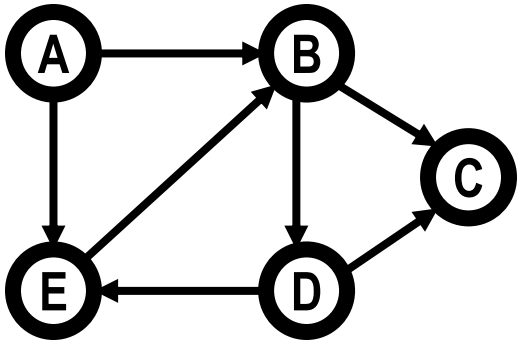
Example: An Adjacency Matrix-based Representation



V	
0	A
1	B
2	C
3	D
4	E
.	⋮
.	
.	

E	0	1	2	3	4	. . .
0	0	1	0	0	1	
1	1	0	1	1	1	
2	0	1	0	1	0	
3	0	1	1	0	1	
4	1	1	0	1	0	
.				⋮		
.						
.						

Example: An Adjacency Matrix-based Representation



V

0	A
1	B
2	C
3	D
4	E
.	⋮
.	
.	

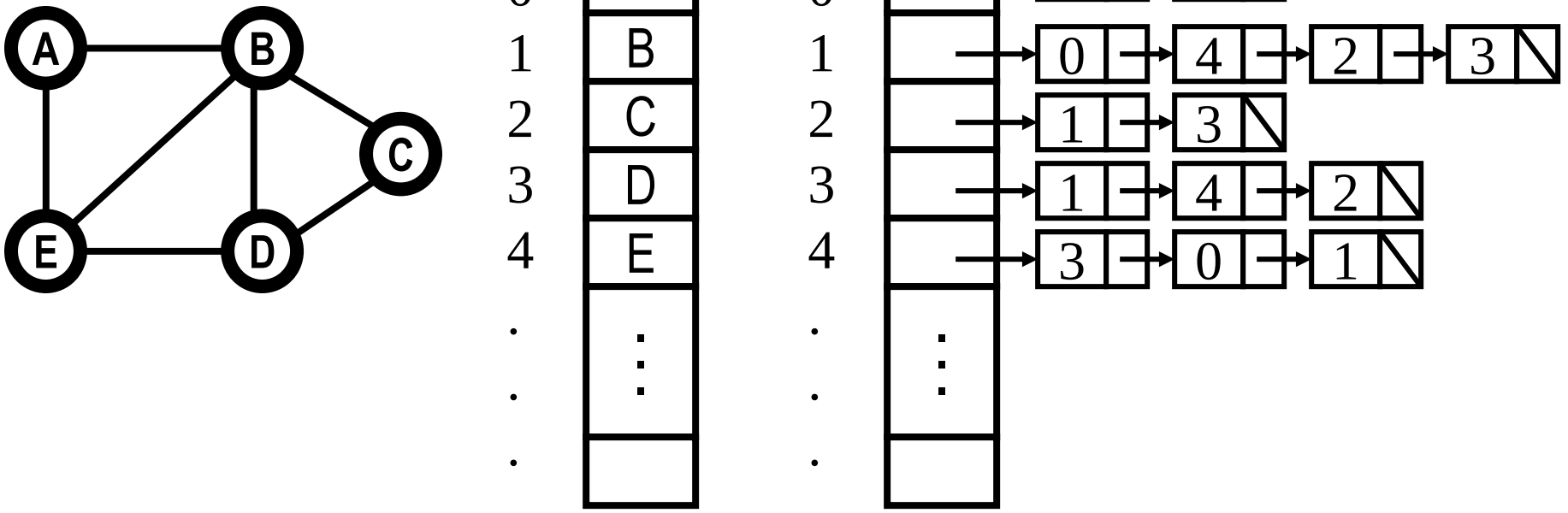
E

	0 1 2 3 4 . . .
0	0 1 0 0 1
1	0 0 1 1 0
2	0 0 0 0 0
3	0 0 1 0 1
4	0 1 0 0 0
.	⋮
.	
.	

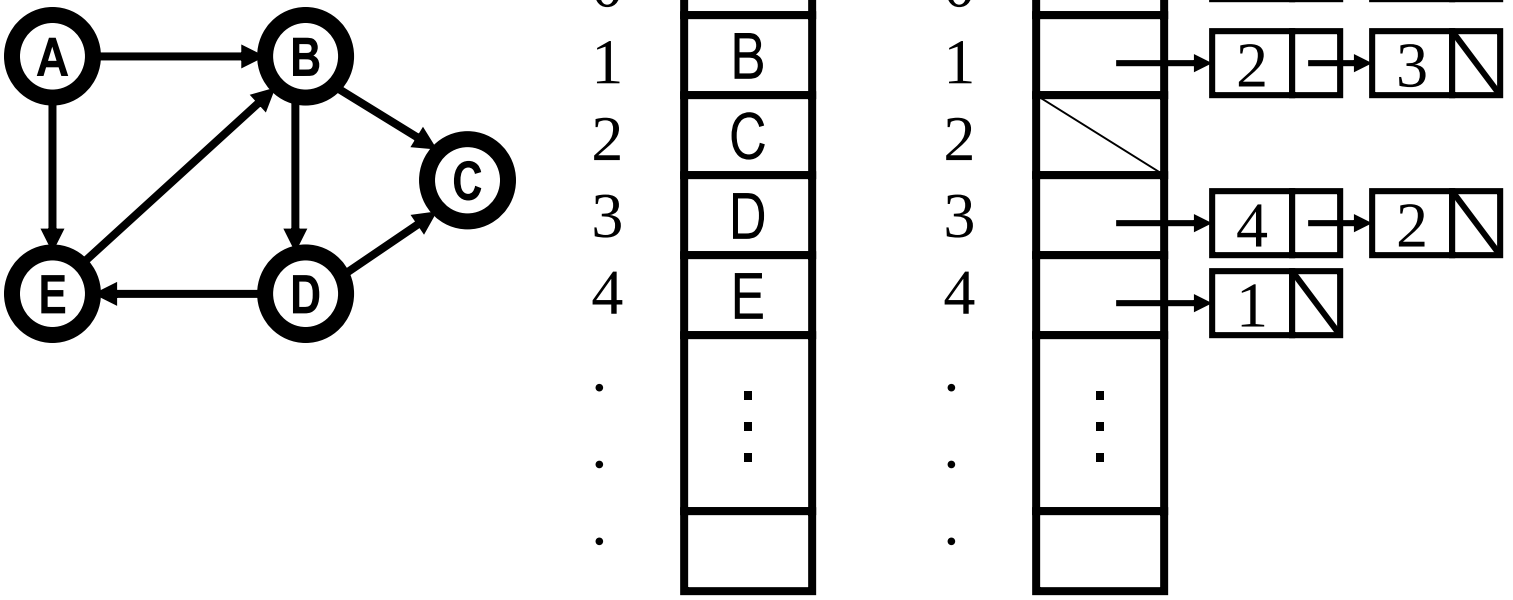
2. An Adjacency (linked) List-based Representation

- Use a linked-list.
- For an undirected or directed graph G = (V, E)
  - ➔ Use |V| linked lists - one for each vertex.
  - ➔ The linked list for vertex vi in V contains the set of vertices adjacent to vi.
- For an edge-weighted undirected or directed graph G = (V, E)
  - ➔ Use |V| linked lists - one for each vertex.
  - ➔ The linked list for vertex vi in V contains the set of vertices adjacent to vi and weights (numeric labels).

Example: An Adjacency (linked) List-based Representation



Example: An Adjacency (linked) List-based Representation



Comparison of Graph Representations

- Space comparison
- Time comparison

1. Space Comparison of Graph Representations

- Adjacent matrix
  - ➔ $O(|V|^2)$ space
- Adjacent list
  - ➔ $O(|V| + |E|)$ space

2. Time Comparison of Graph Representations

- Operation: **Adding or removing an edge.**
  - Adjacent matrix
    - O(1)
    - O(1)
  - Adjacent list
    - O(1)
    - O(|E|)

Time Comparison of Graph Representations

- Operation: **Determine whether there is an edge from vertex u to another vertex v.**
  - Adjacent matrix
    - O(1) time
  - Adjacent list
    - O(|E|) time

Time Comparison of Graph Representations

- Operation: **Find all edges.**
  - Adjacent matrix
    - $O(|V|^2)$ time
  - Adjacent list
    - $O(|V|+|E|)$ time

Time Comparison of Graph Representations

- Operation: **Find all vertices v that are adjacent to a given vertex u s.t. (u,v).**
  - Adjacent matrix
    - O(|V|)
  - Adjacent list
    - O(|E|)

Time Comparison of Graph Representations

- Operation: **Find all vertices u that a given vertex v is adjacent to (u,v).**
  - Adjacent matrix
    - O(|V|) time
  - Adjacent list
    - O(|V|+|E|) time

►QUIZ?

- Adjacent matrix vs. Adjacent list?

Operations on Graphs

Graphs Traversals

- Often it is useful to visit the vertices of a graph in some specific order.
- Two common ways:
  - **Depth-first search/traversal**
  - **Breadth-first search/traversal**

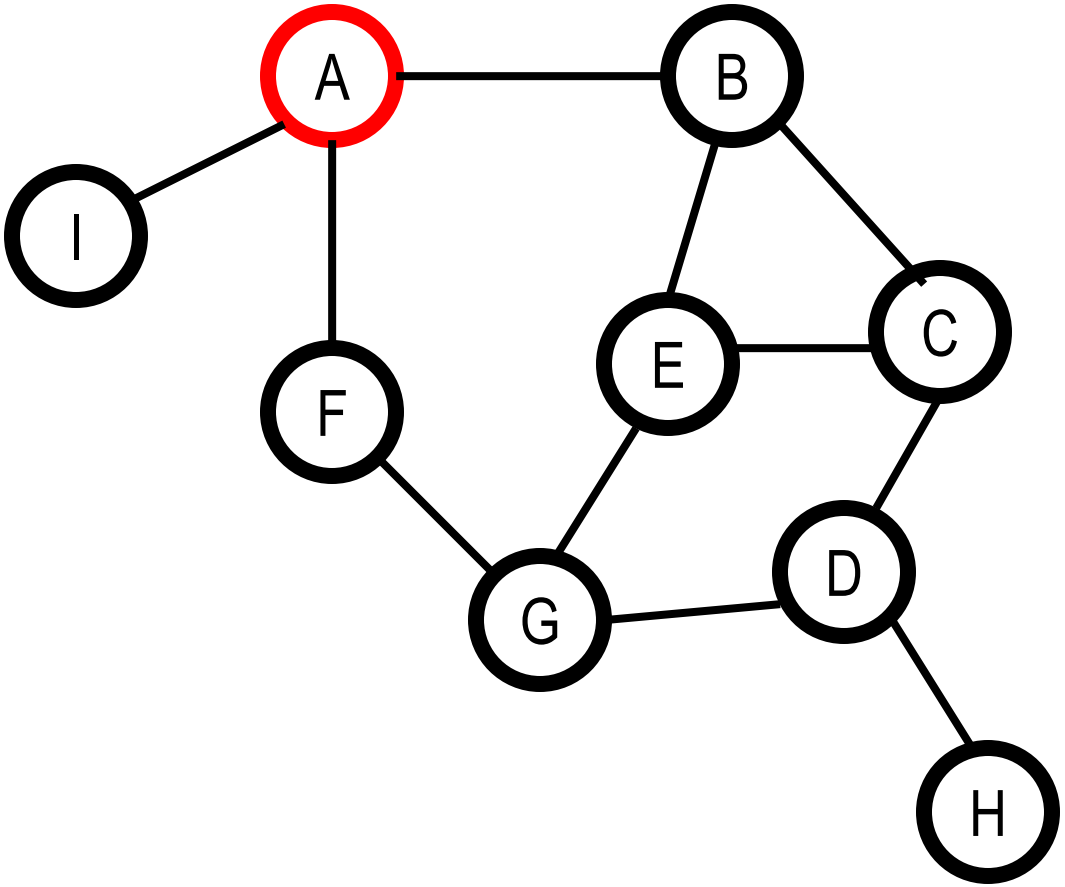
Tree Traversal Vs Graph Traversal?

- Tree
  - ➔Acyclic
- Graphs
  - ➔Could have cycles!
  - ➔Unlike tree traversal, need to specifically guard against repeating a path from a cycle.
  - ➔Mark each vertex as “visited” when we encounter it and not consider visited vertices more than once.

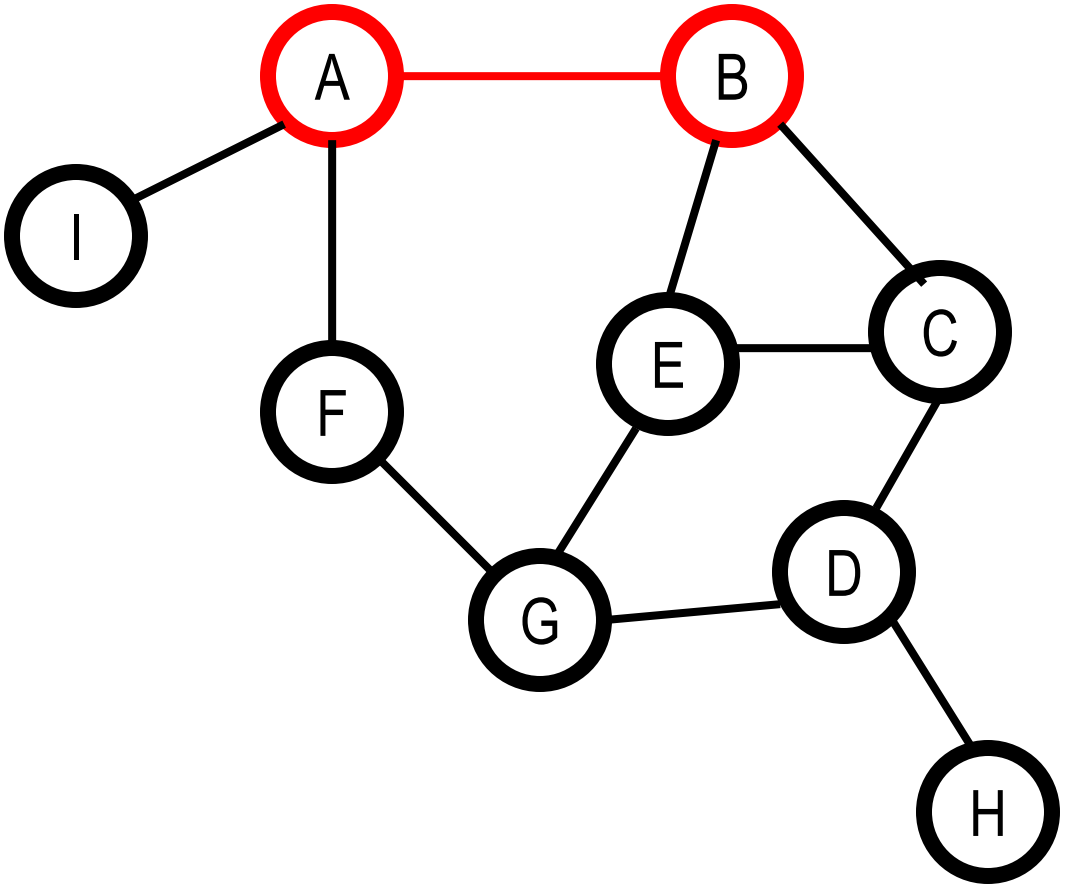
Depth-First Search (DFS) - Recursive

- DFS(u):
  - ➔ Mark u as visited;
  - ➔ for each **unvisited** vertex v **adjacent** to u (i.e., edge (u,v)) do
    - ☞ DFS(v)

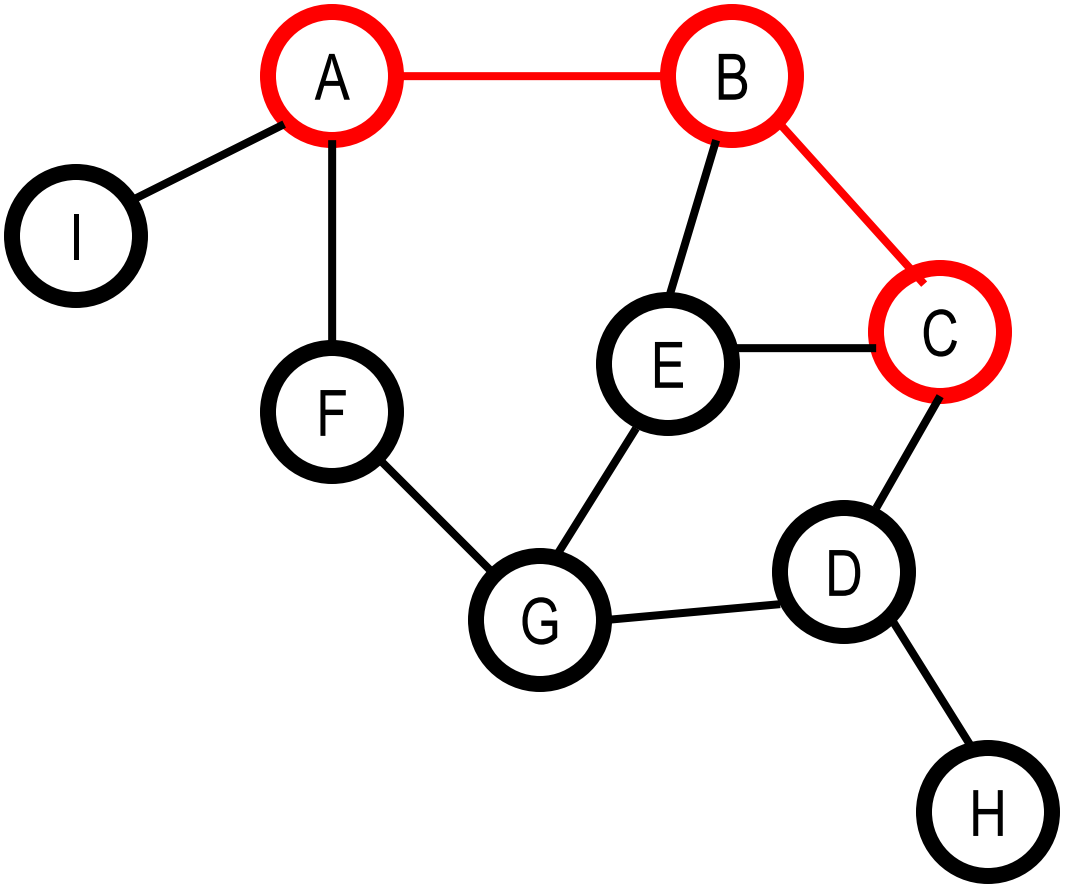
Example: DFS(A) - Recursive - of a connected, undirected graph



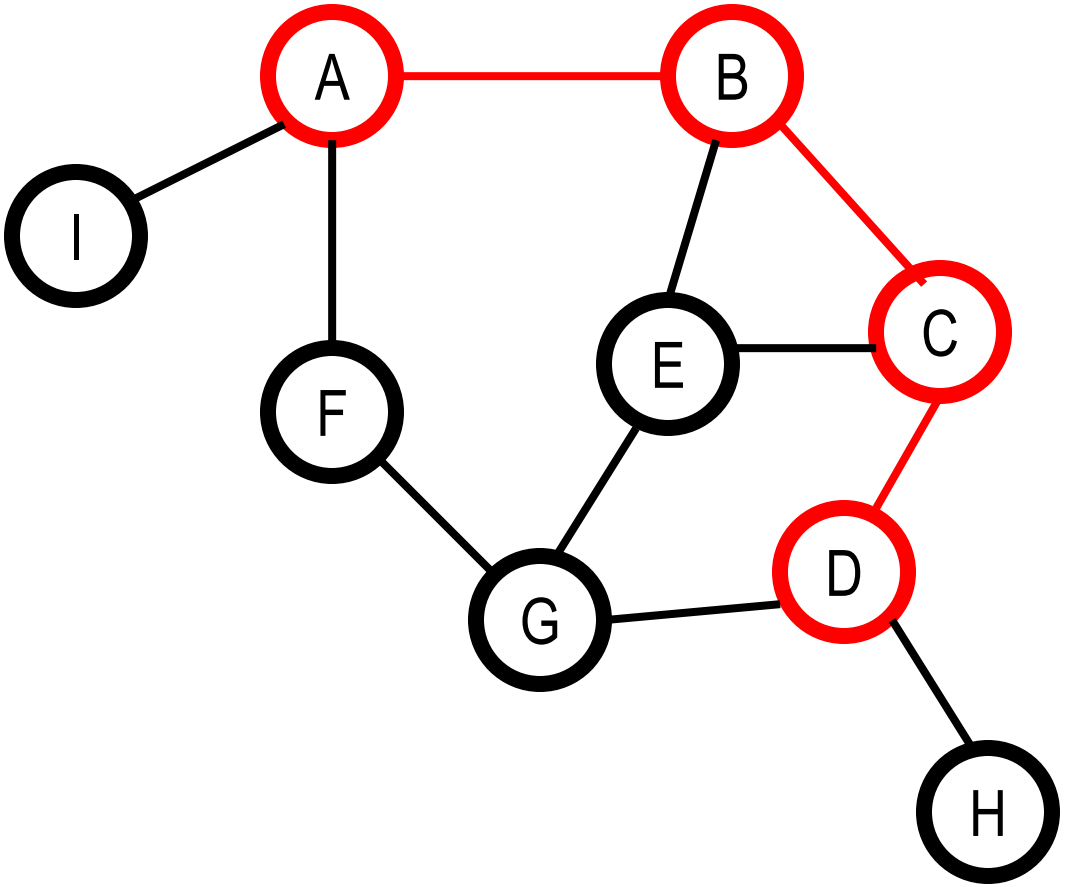
Example: DFS(A)



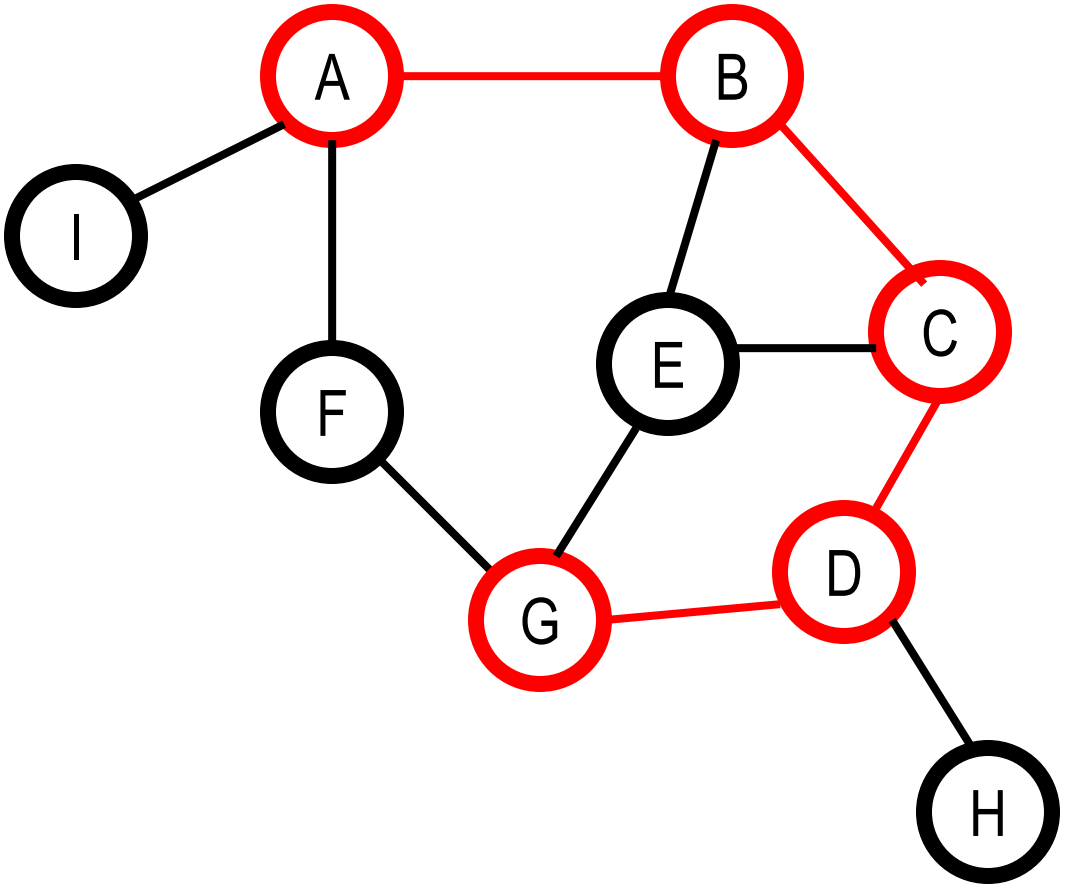
Example: DFS(A)



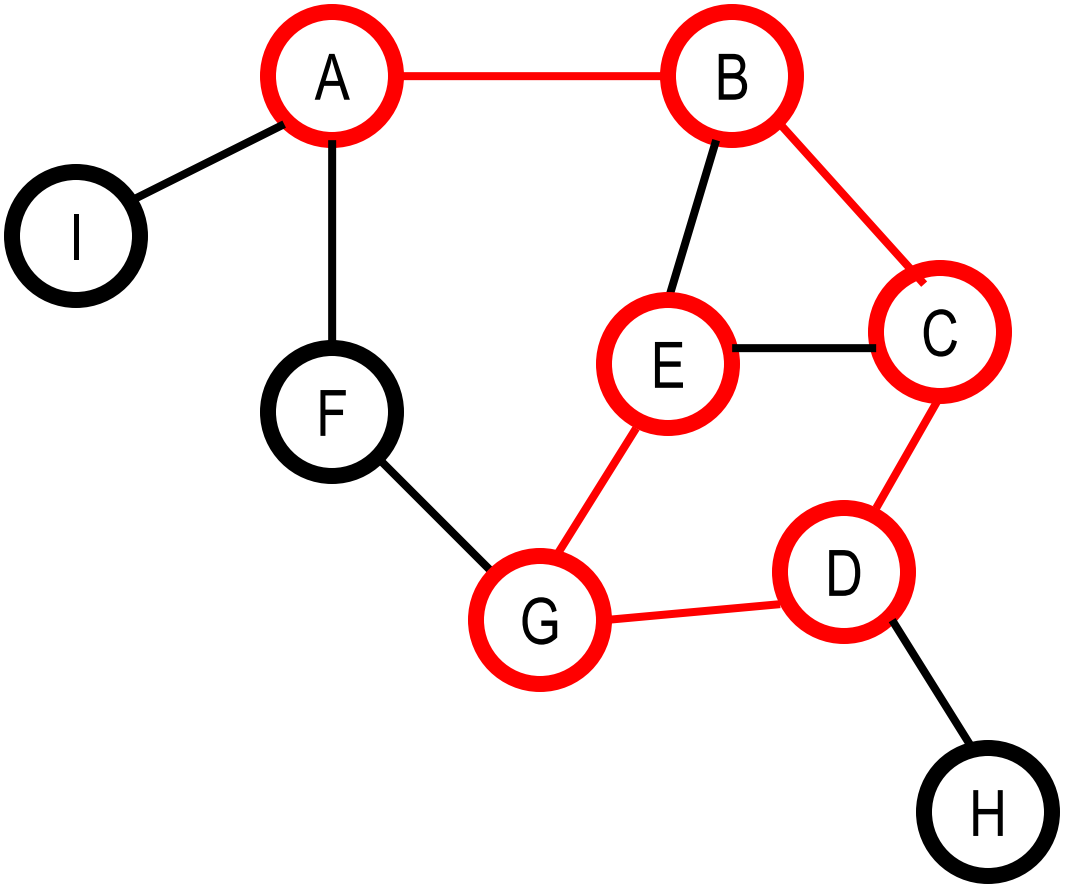
Example: DFS(A)



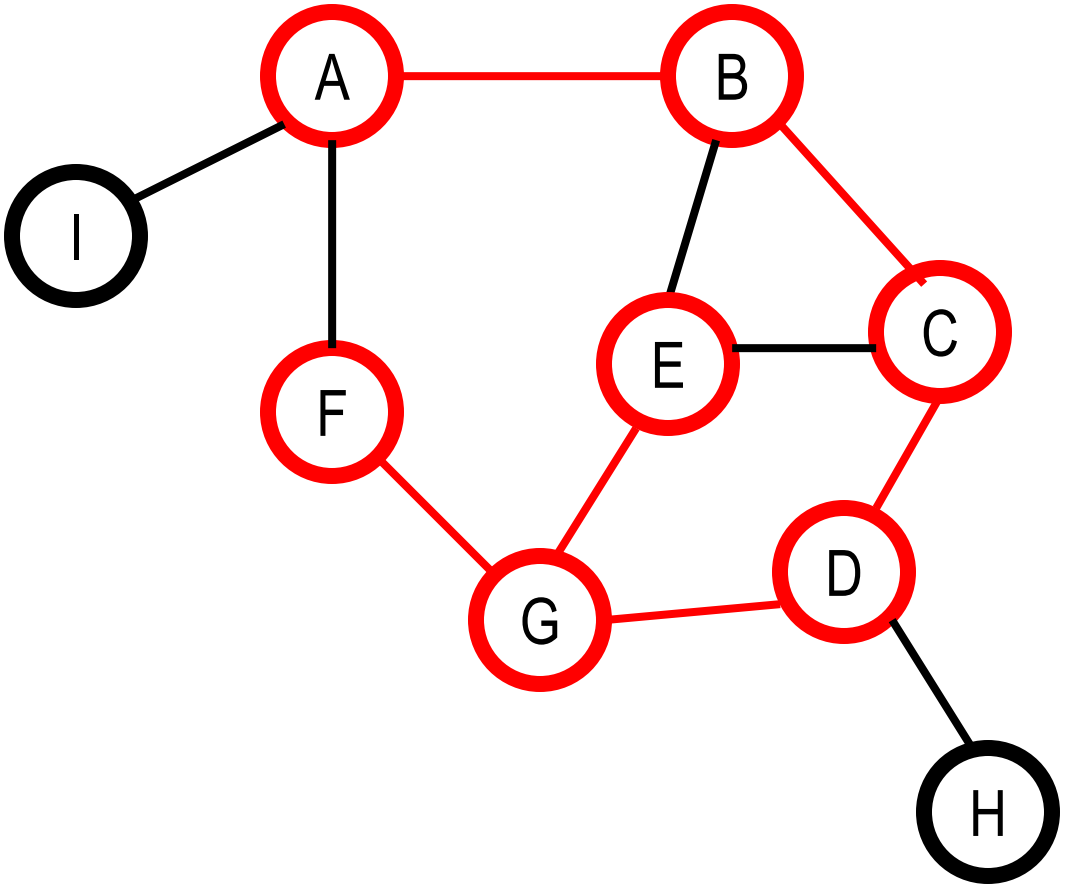
Example: DFS(A)



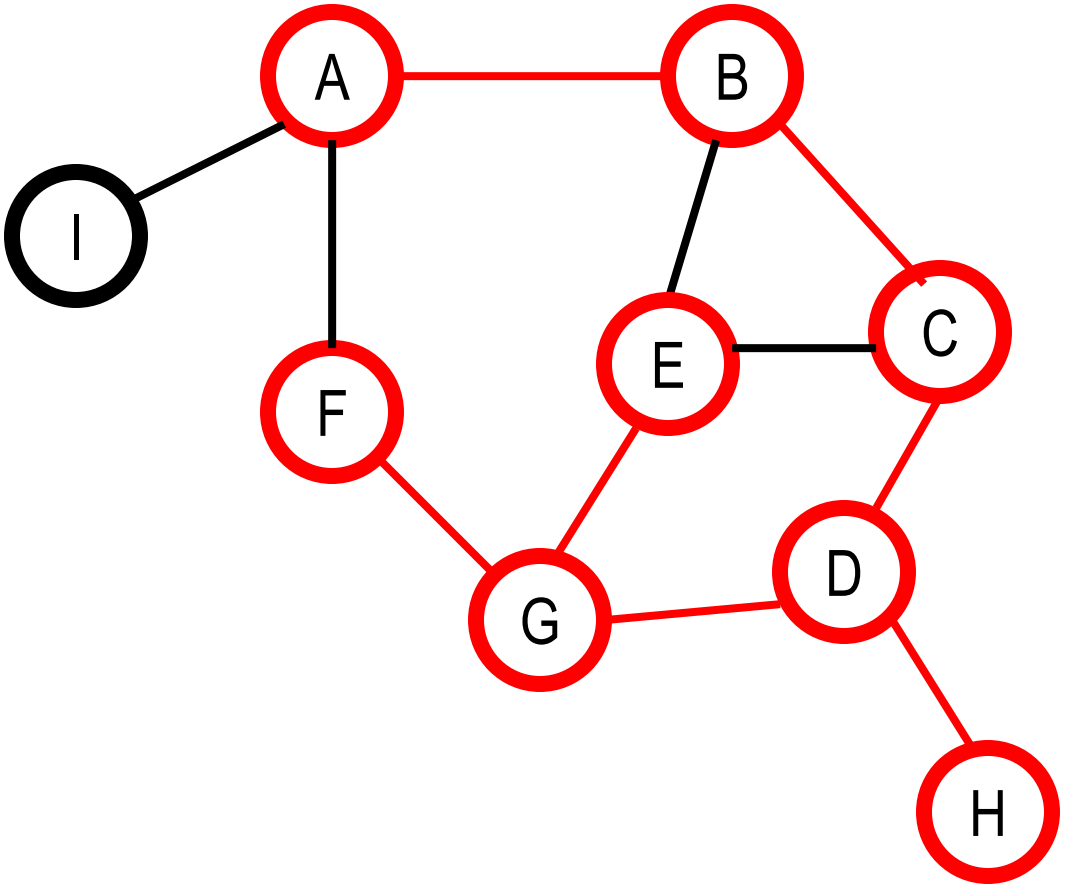
Example: DFS(A)



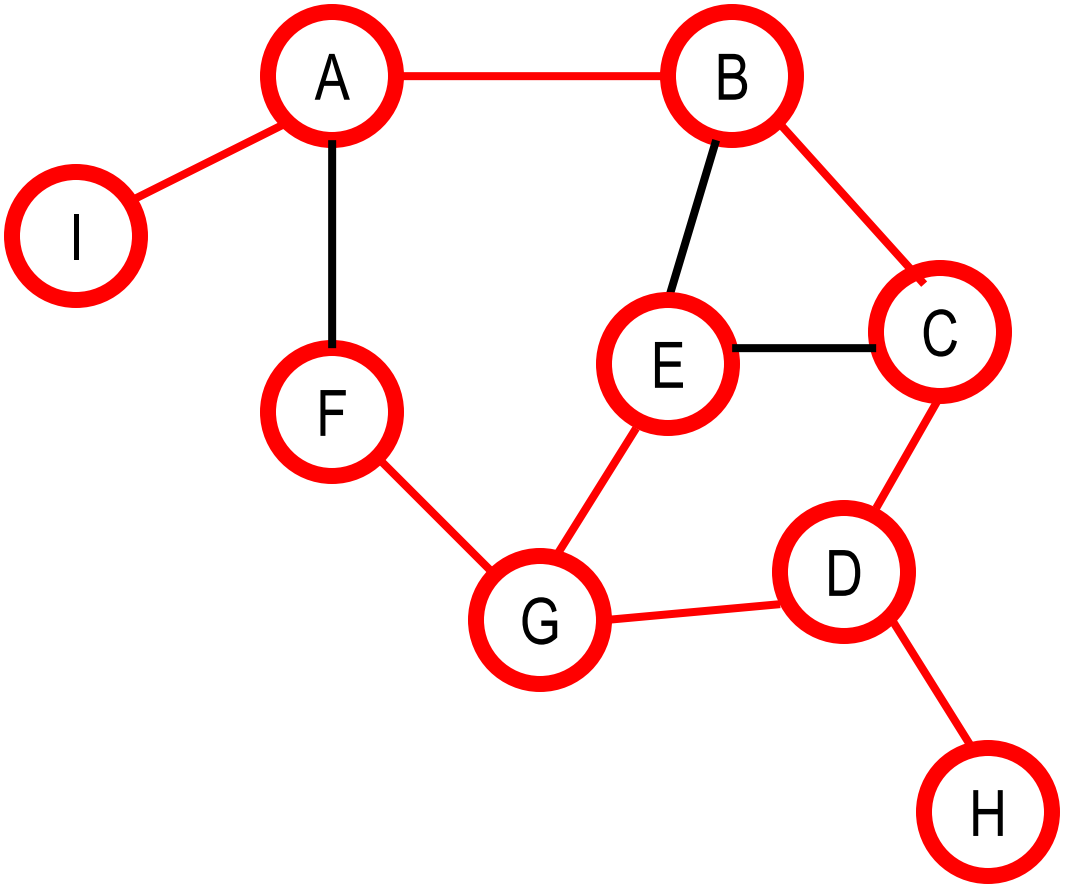
Example: DFS(A)



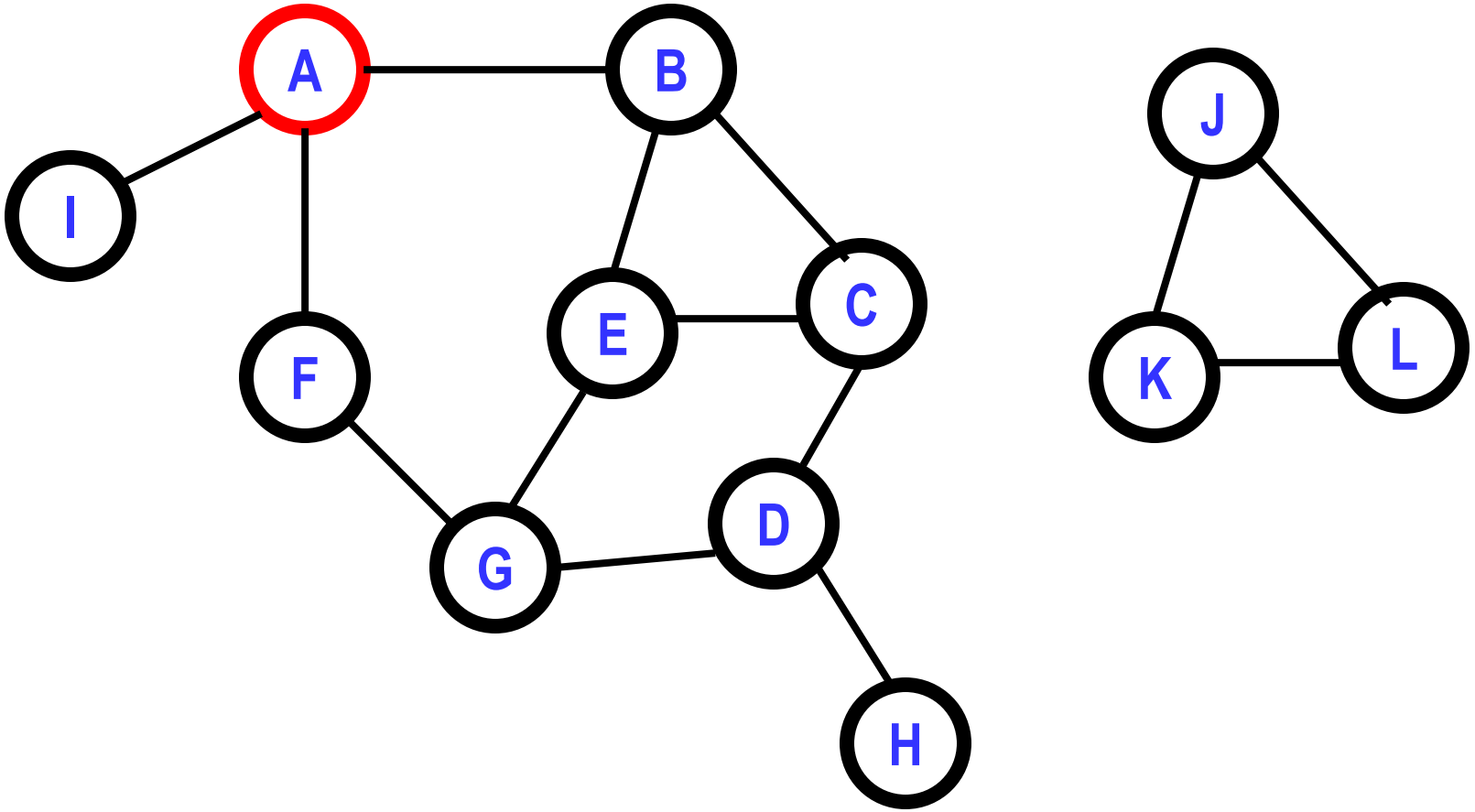
Example: DFS(A)



Example: DFS(A)



▶QUIZ?



Depth-First Search (DFS) - Analysis

- Worst-case running time
  - Using adjacent matrix
    - $O(|V|^2)$ time
  - Using adjacent list
    - $O(|V|+|E|)$ time

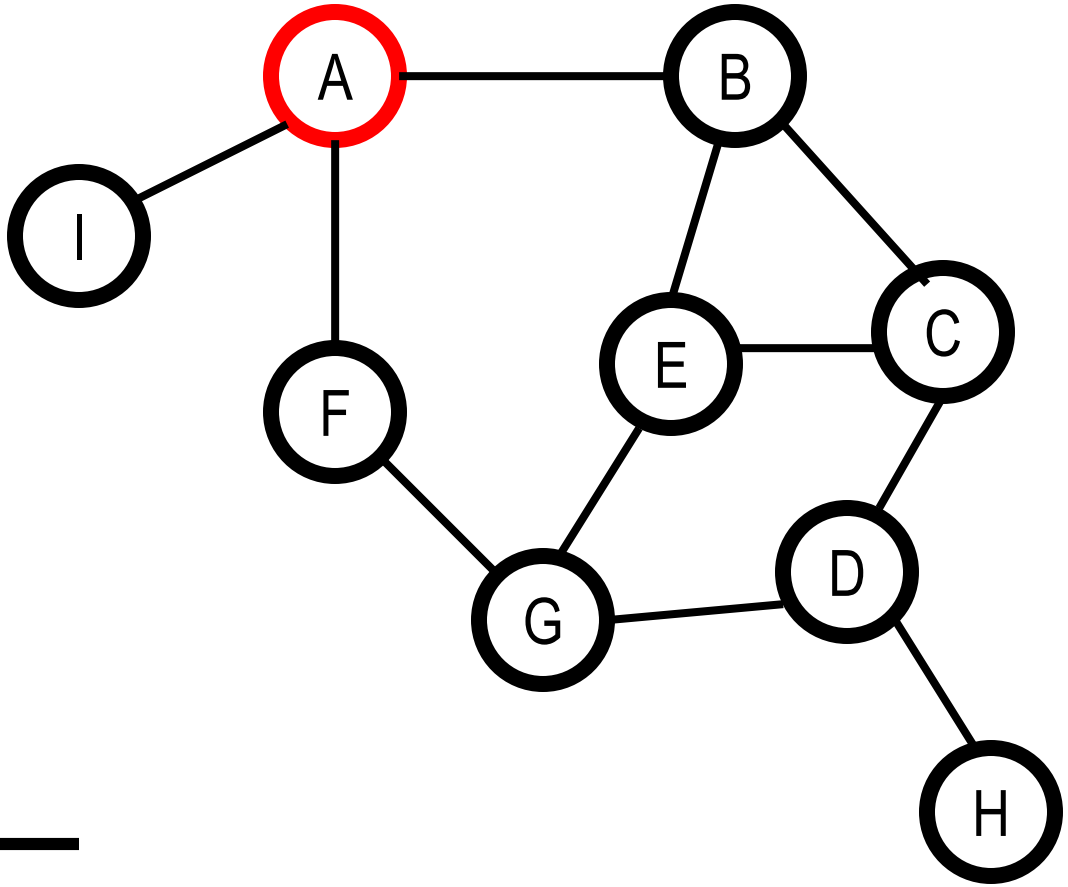
Depth-First Search (DFS) Visualization

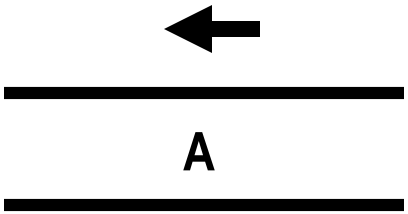
- *Depth-First Search (DFS) Visualization*

Breadth-First Search (BFS) – Iterative using Queue

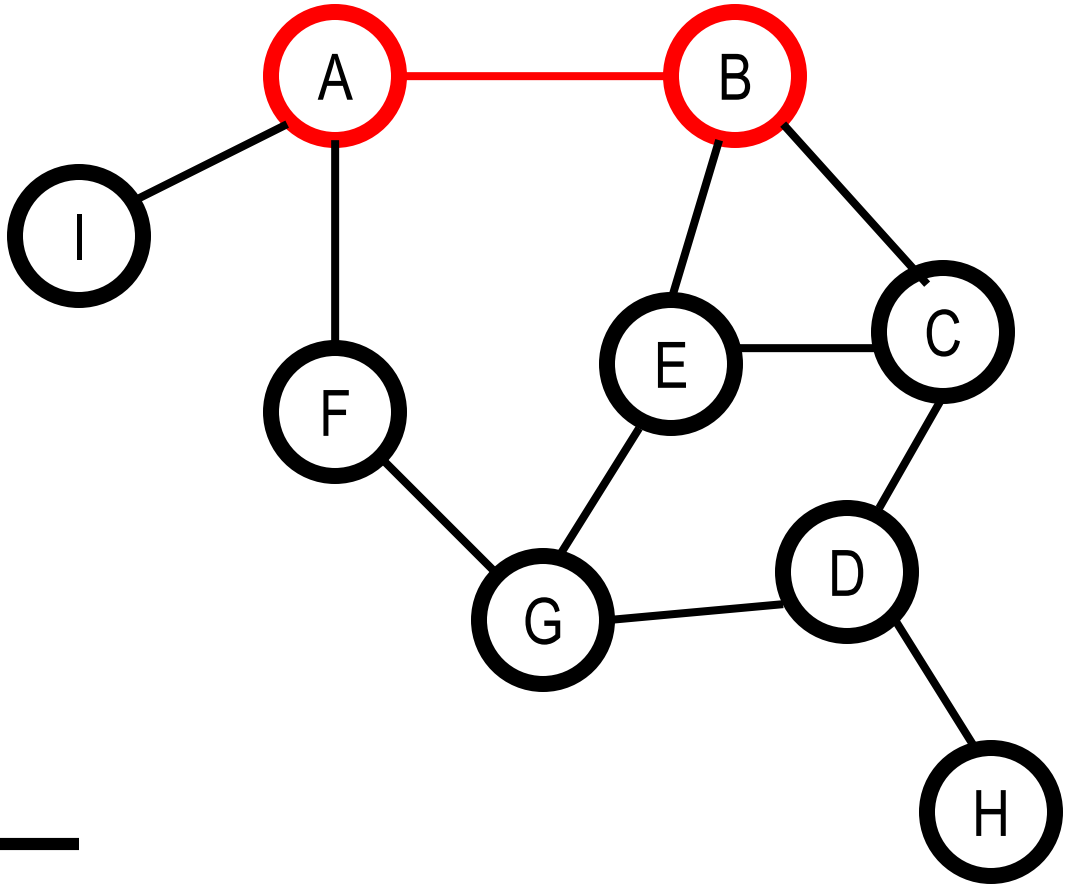
- BFS(u):
  - Q.CreateQueue();
  - Q.Insert(u); Mark u as visited;
  - while !Q.QueueEmpty() do
    - Q.QueueDelete(w);
    - for each **unvisited** vertex v adjacent to w do
      - Mark v as visited;
      - Q.QueueInsert(v);

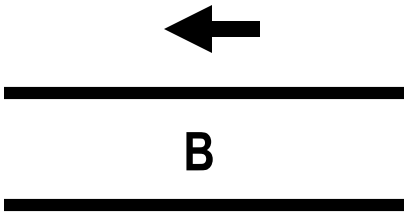
Example: BFS(A) - Iterative



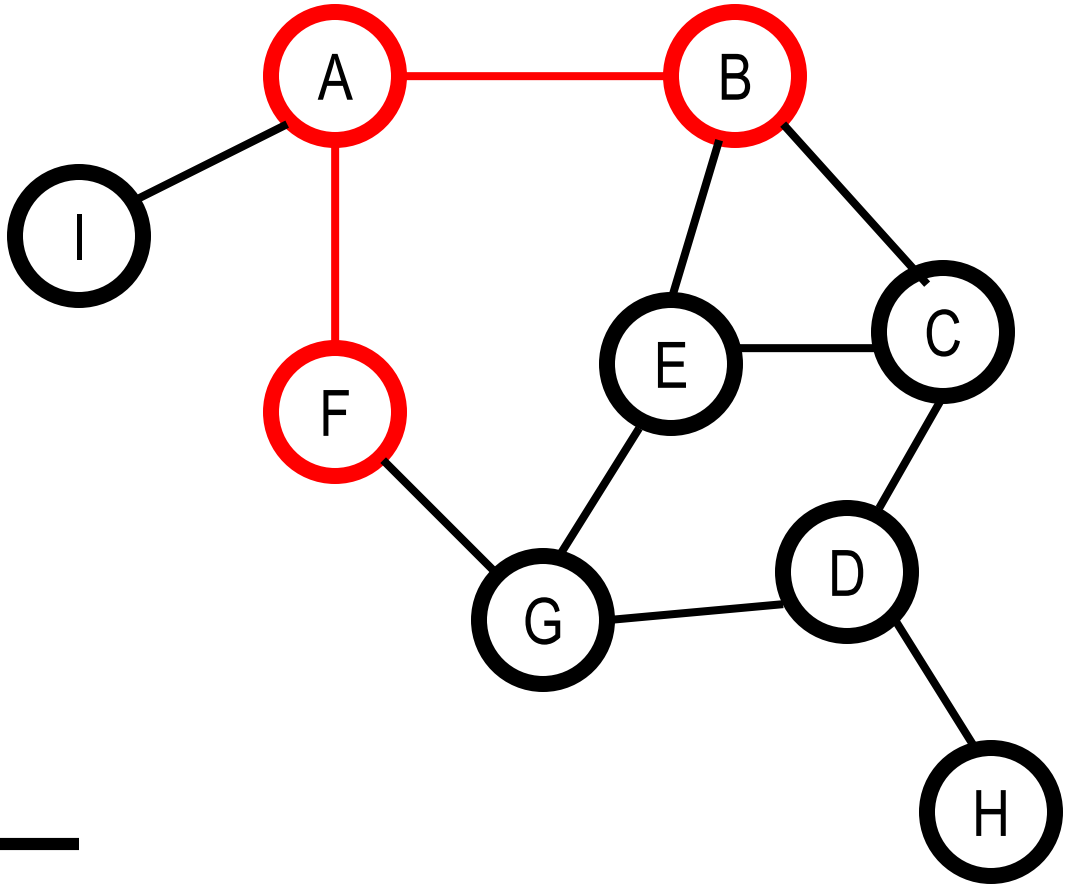


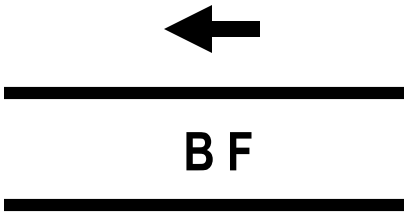
Example: BFS(A) - Iterative



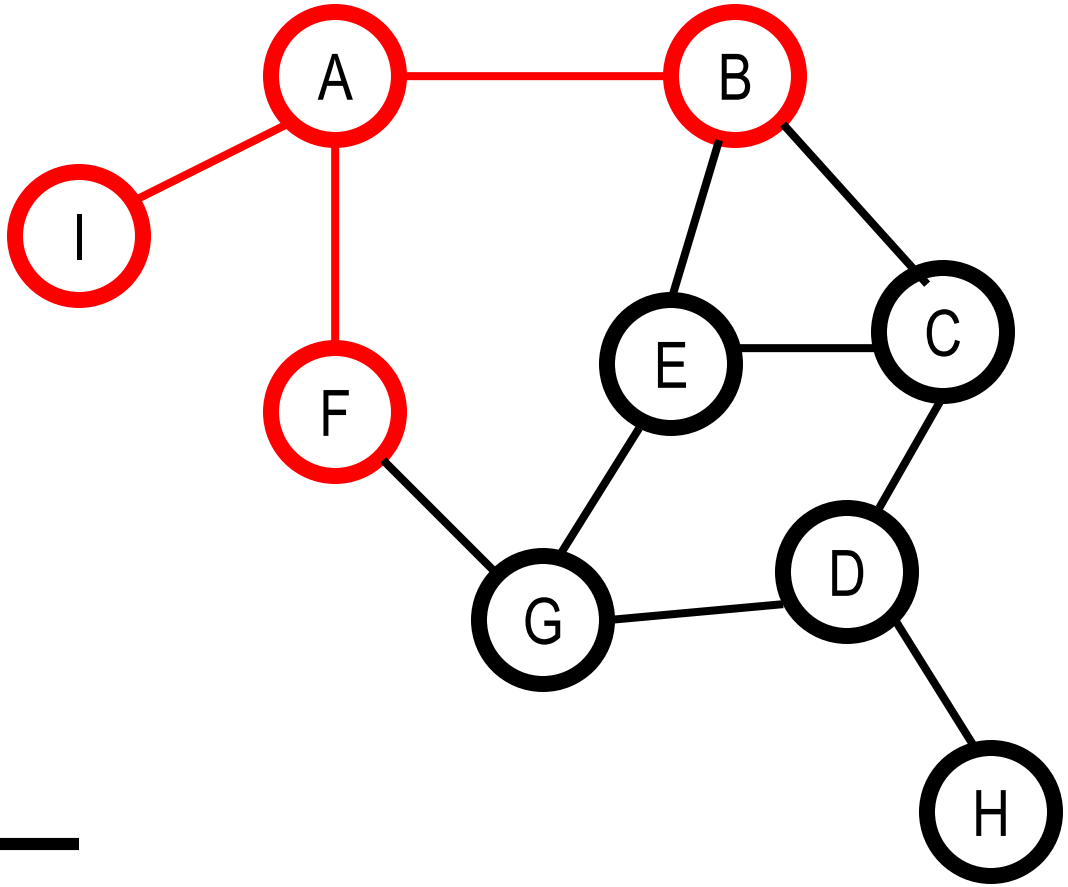


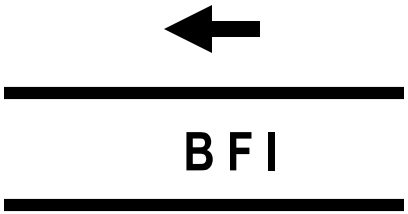
Example: BFS(A) - Iterative



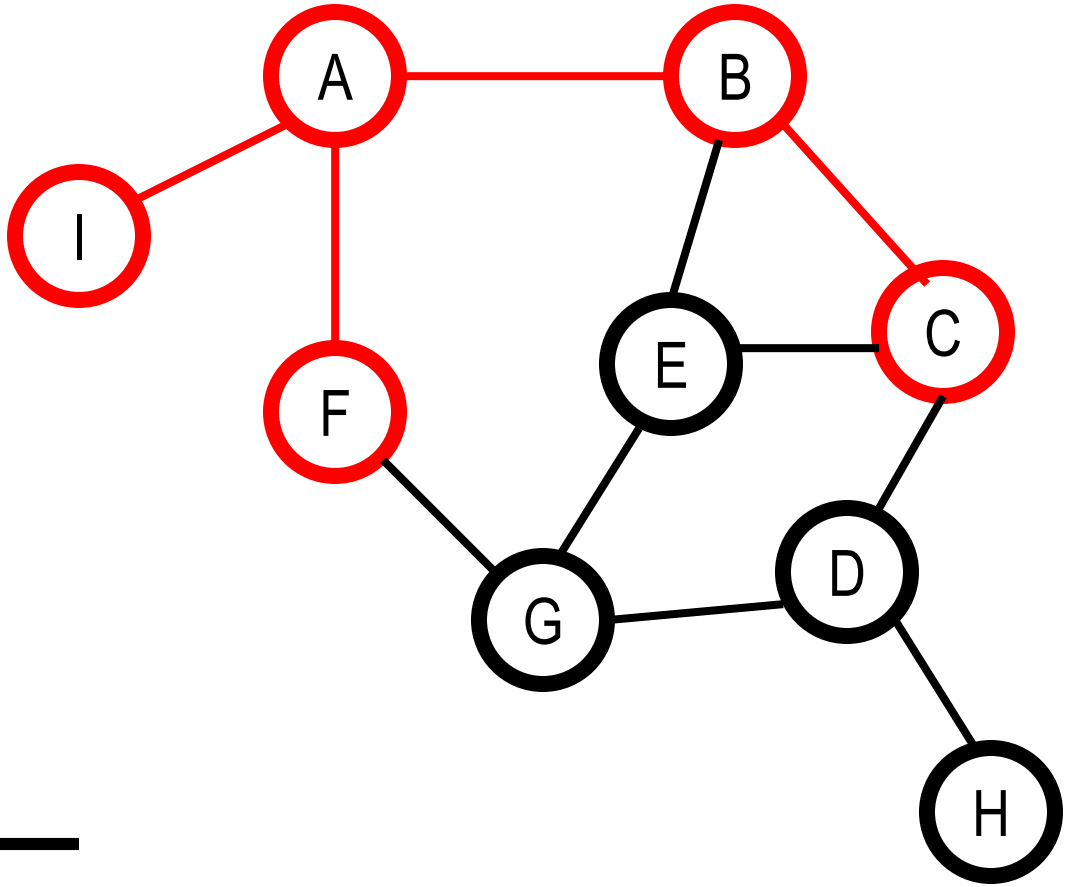


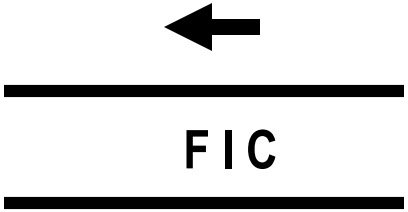
Example: BFS(A) - Iterative



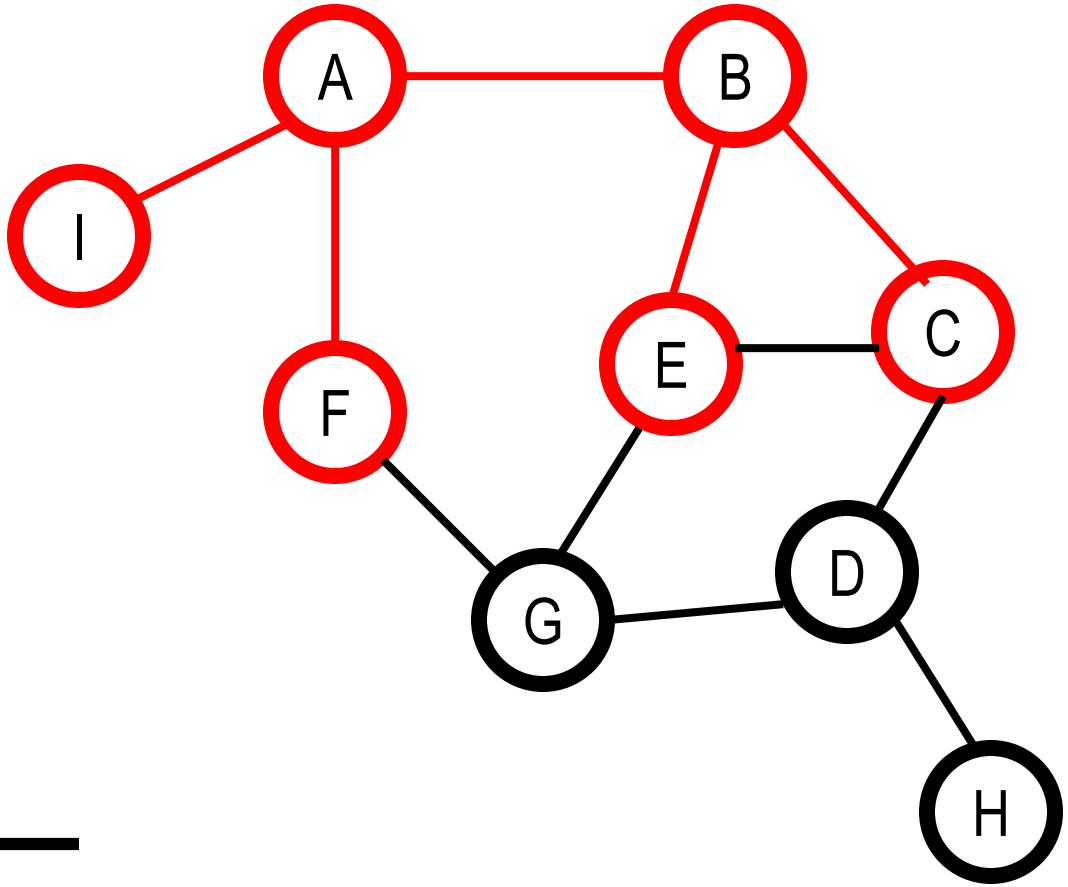


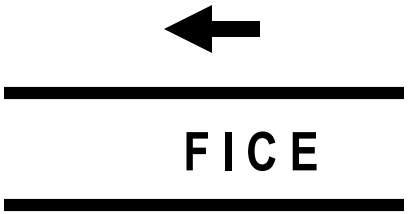
Example: BFS(A) - Iterative



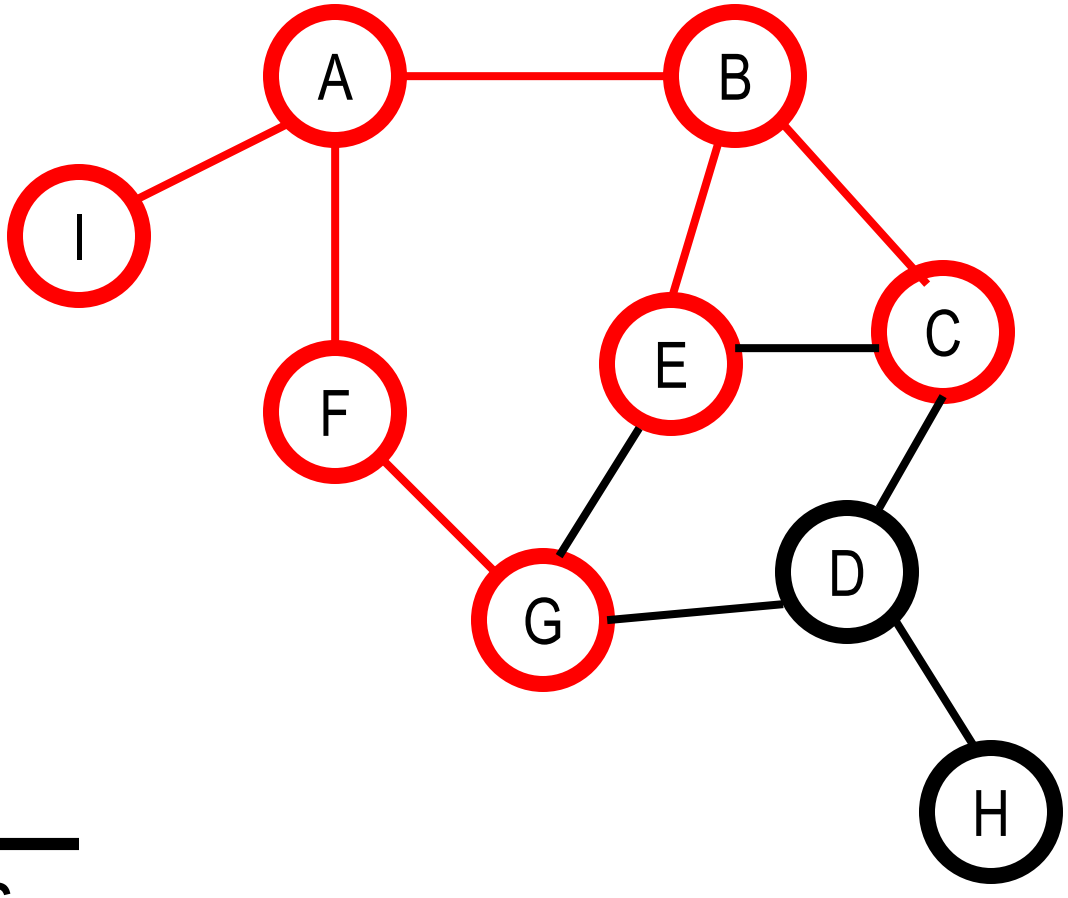


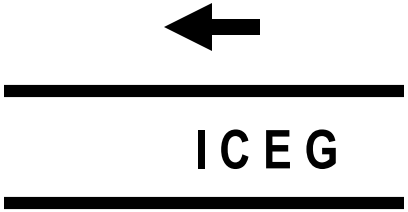
Example: BFS(A) - Iterative



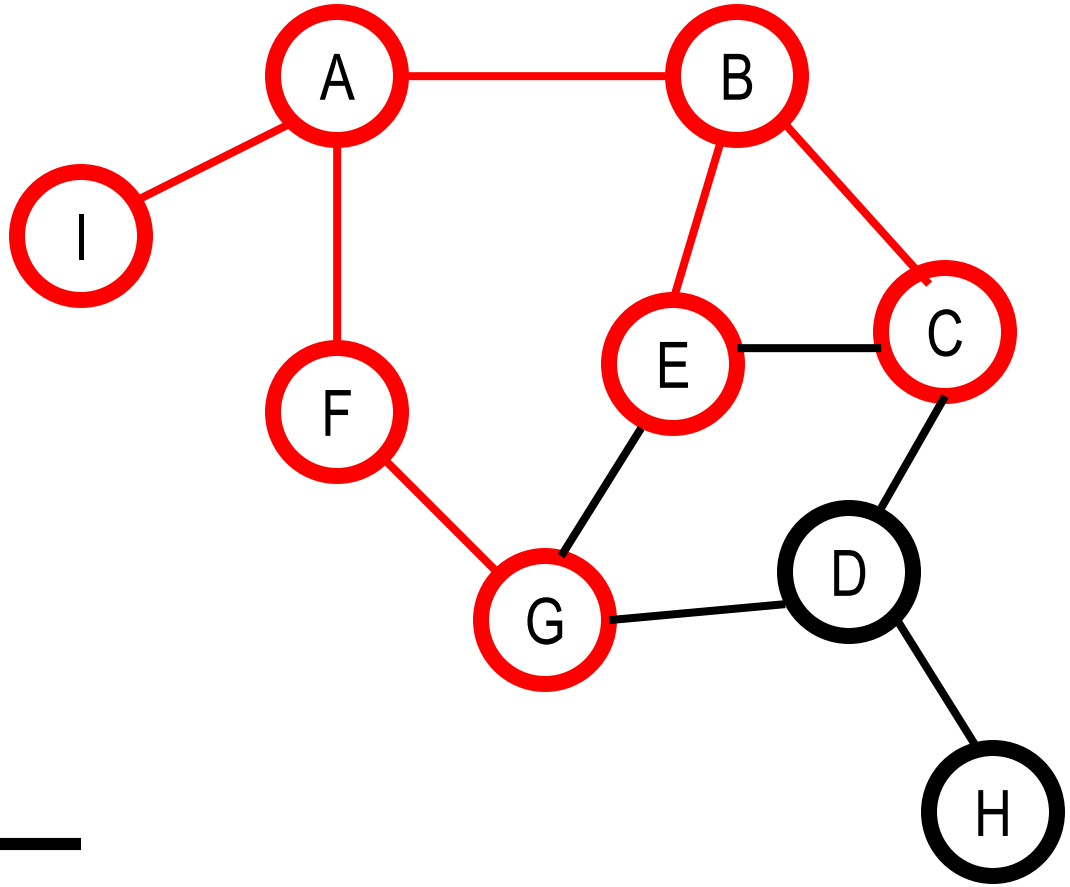


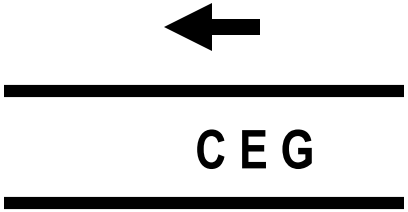
Example: BFS(A) - Iterative



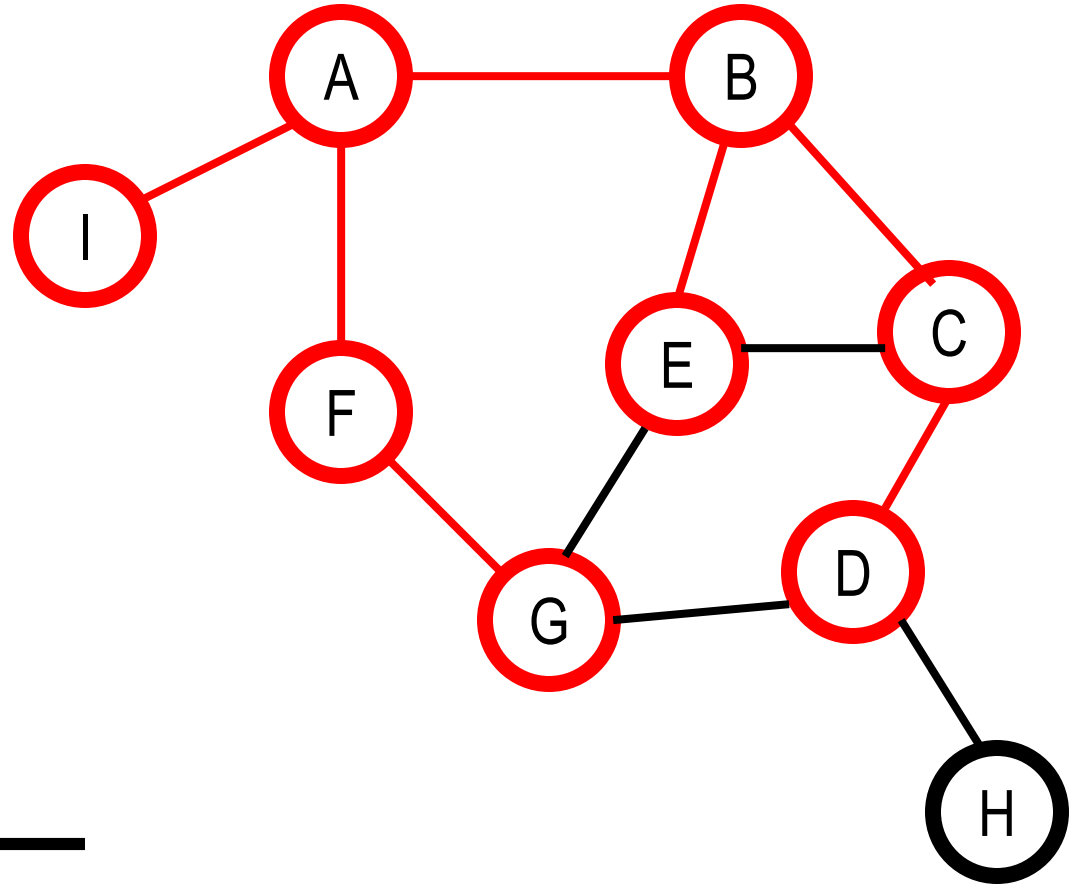


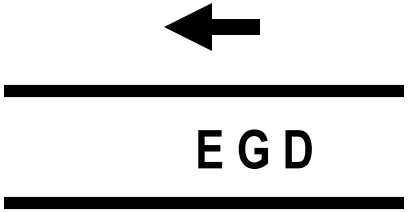
Example: BFS(A) - Iterative



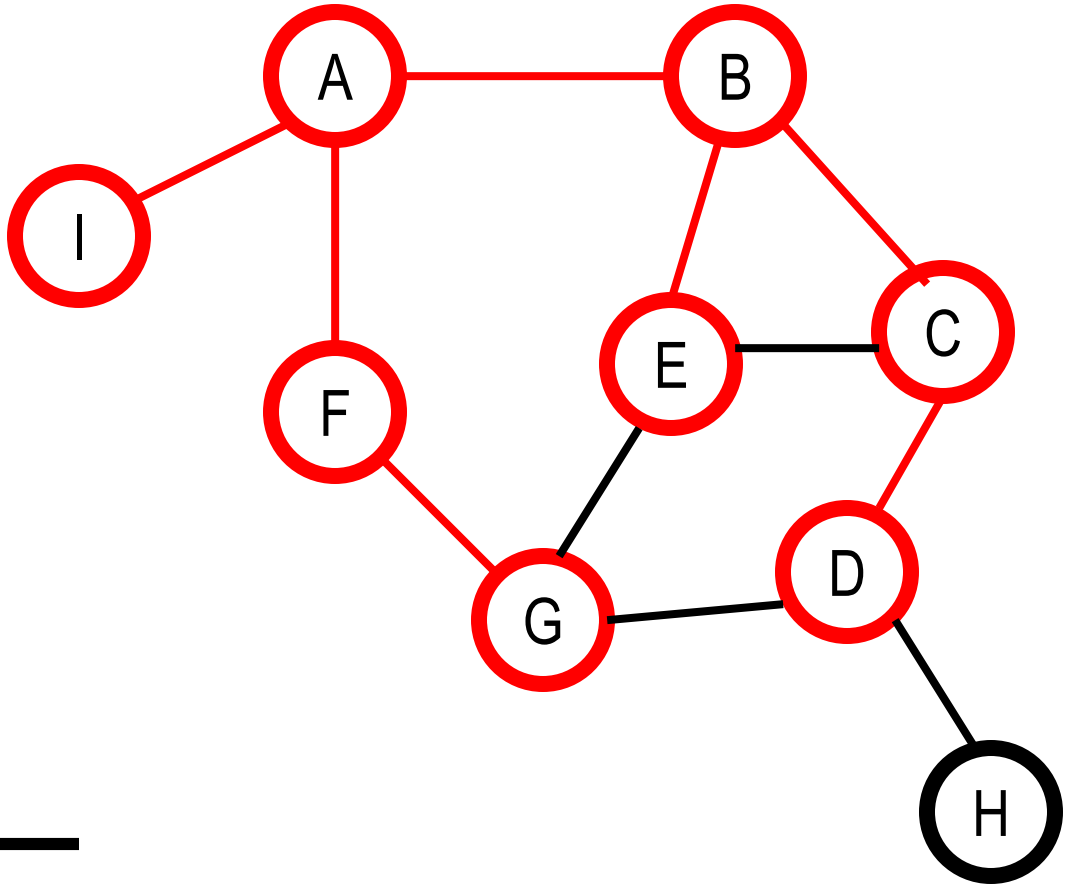


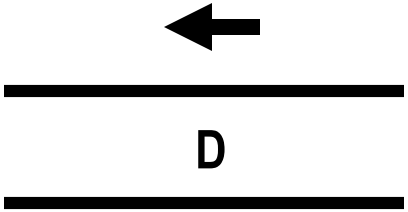
Example: BFS(A) - Iterative



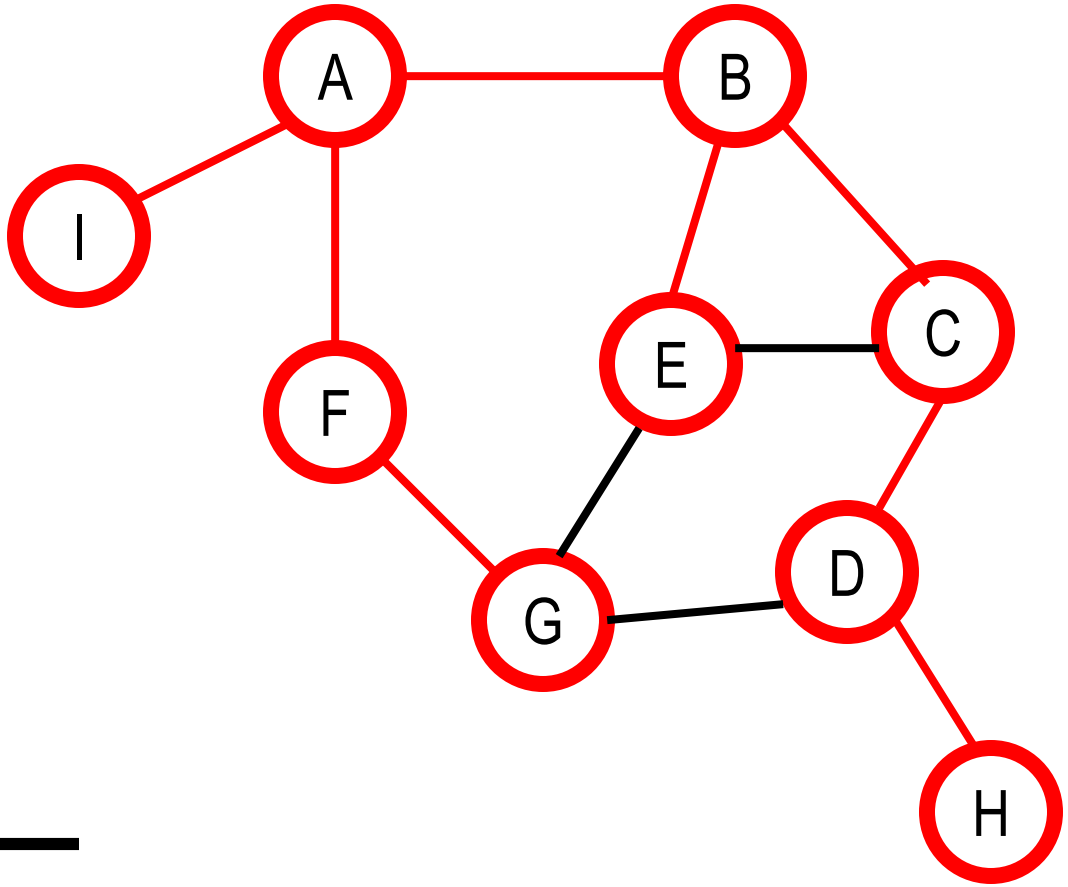


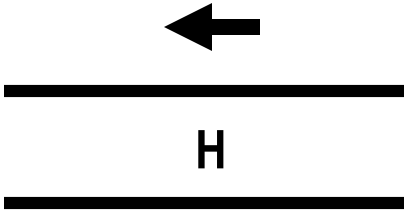
Example: BFS(A) - Iterative



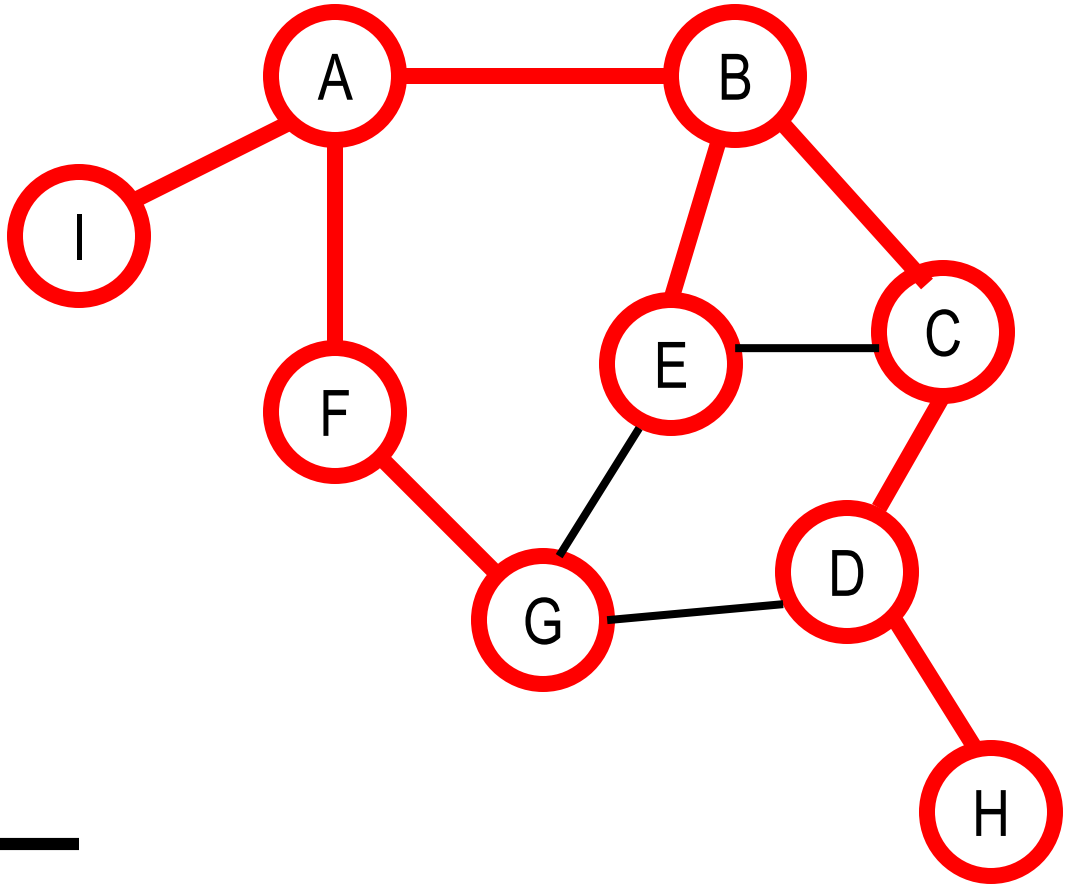


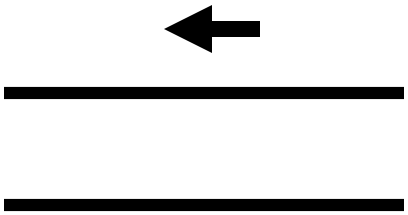
Example: BFS(A) - Iterative



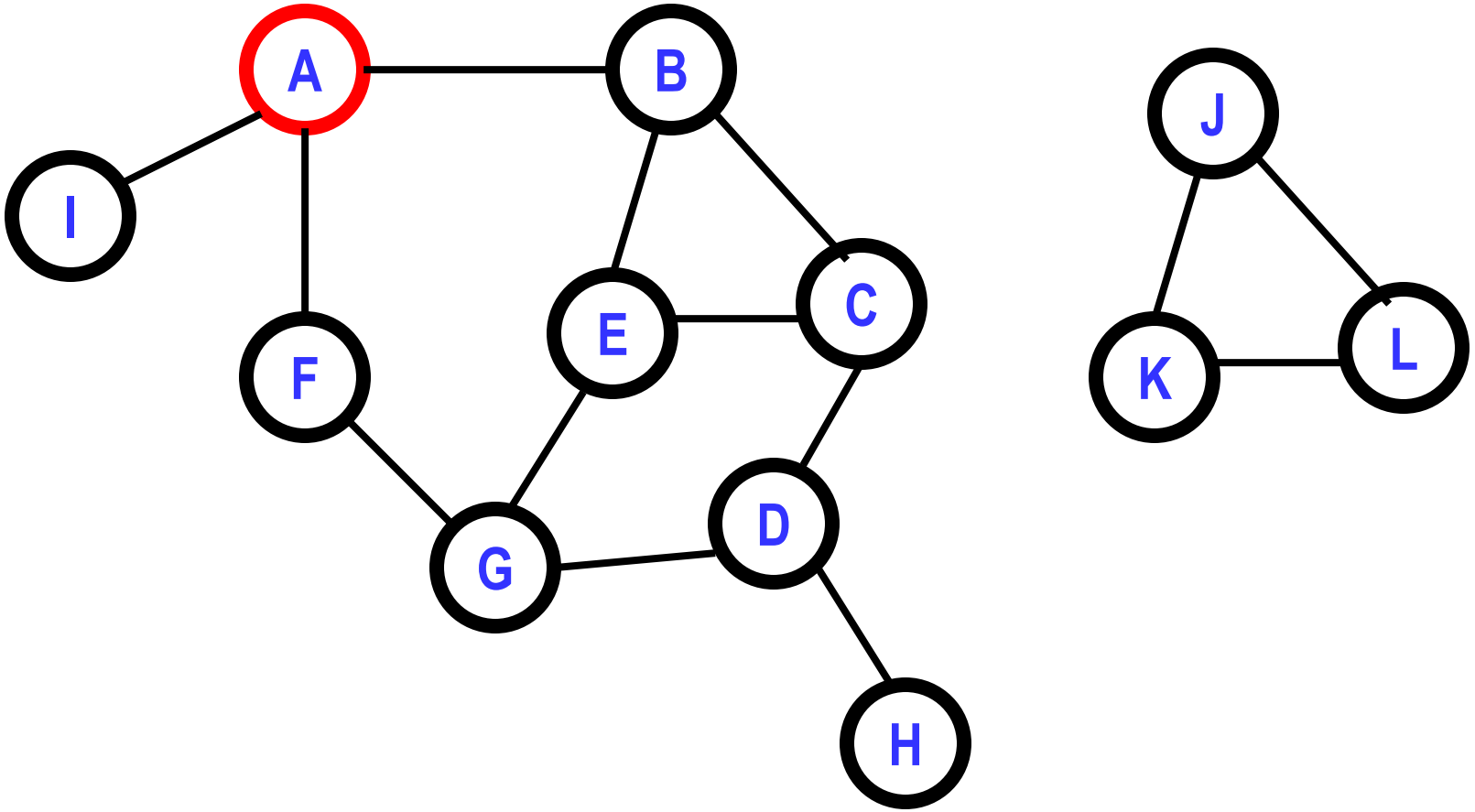


Example: BFS(A) - Iterative





▶QUIZ?



Breadth-First Search (BFS) - Analysis

- Worst-case running time
  - Using adjacent matrix
    - $O(|V|^2)$ time
  - Using adjacent list
    - $O(|V|+|E|)$ time

Breadth-First Search (BFS) Visualization

- *Breadth-First Search (BFS) Visualization*

►QUIZ?

- Compare graph traversal DFS vs. BFS?
- Which data structures are used for DFS and BFS of a graph?
- Best-First Search? (**Branch and Bound**)

Fundamental Problems on Graphs: Finding Shortest Paths

A Shortest-path From U to V in a Weighted, Directed Graph

- Let G=(V,E) be a edge-weighted, directed graph with weight function W: E $\rightarrow$ R mapping edges to real valued weights.
  - The length of a path from u to v is the number of its constituent edges.
  - The weighted length of a path P from u to v is the sum of the weights of its constituent edges.

A Shortest-path From U to V in a Weighted, Directed Graph

- The shortest-path weighted length form u to v is
  - ➔ Min (weights of all paths from u to v) if there is a path from u to v
  - ➔ ∞ otherwise
- A shortest path from u to v is any path with the shortest-path weighted length.

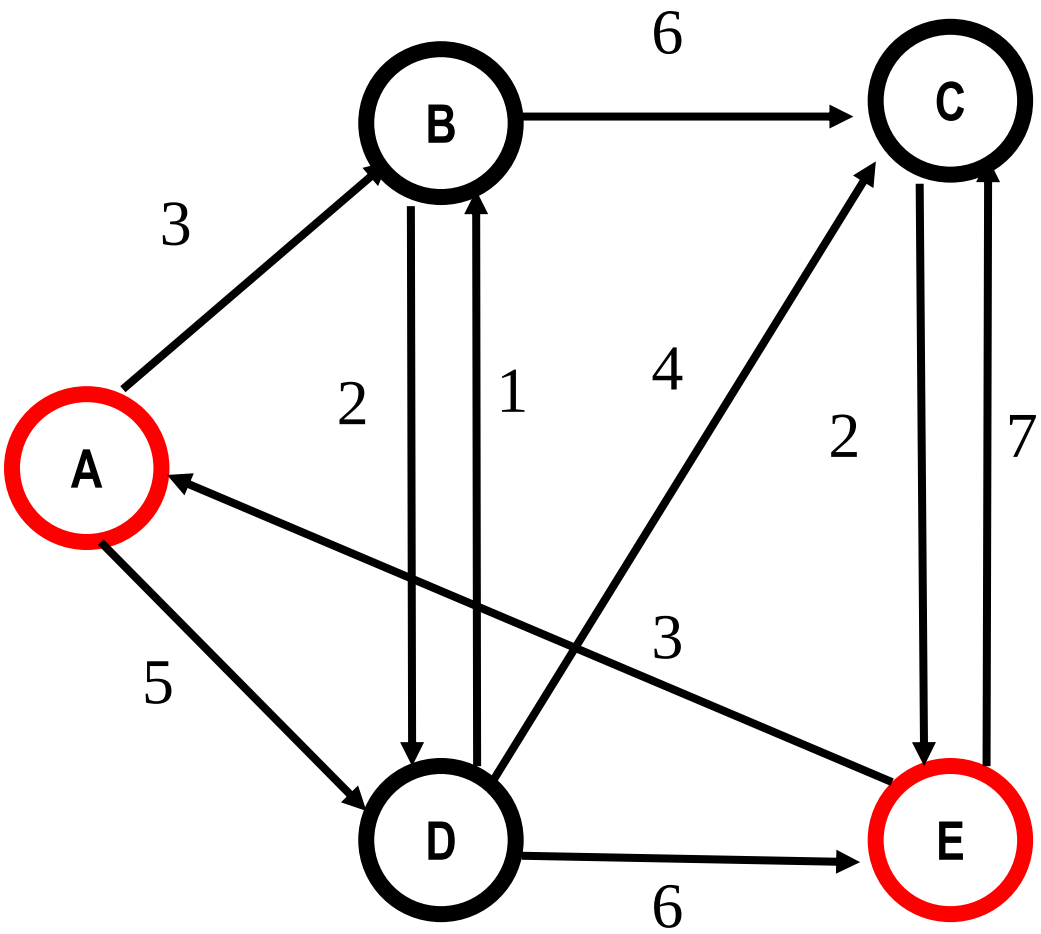
✓ Shortest Path Problems

- **The Single-Pair Shortest Path Problem**
- **The Single-Source Shortest Paths Problem**
- **The Single-Destination Shortest Paths Problem**
- **The All-Pairs Shortest Paths Problem**

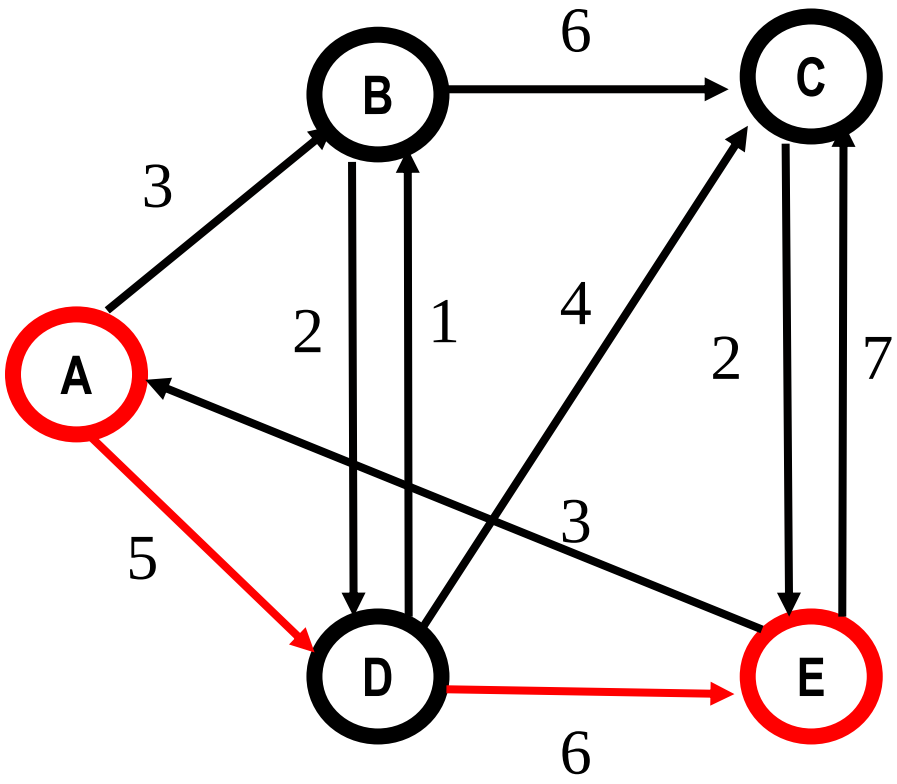
1. The Single-Pair Shortest Path Problem

- Given a weighted, directed graph G=(V,E) with weight function W: E → R mapping edges to real valued weights,
- Find a shortest path from a given source vertex s to a given destination vertex d in V.
  - ➔ The Single-Pair Shortest Path Problem

Example: Shortest-Paths from A to E?

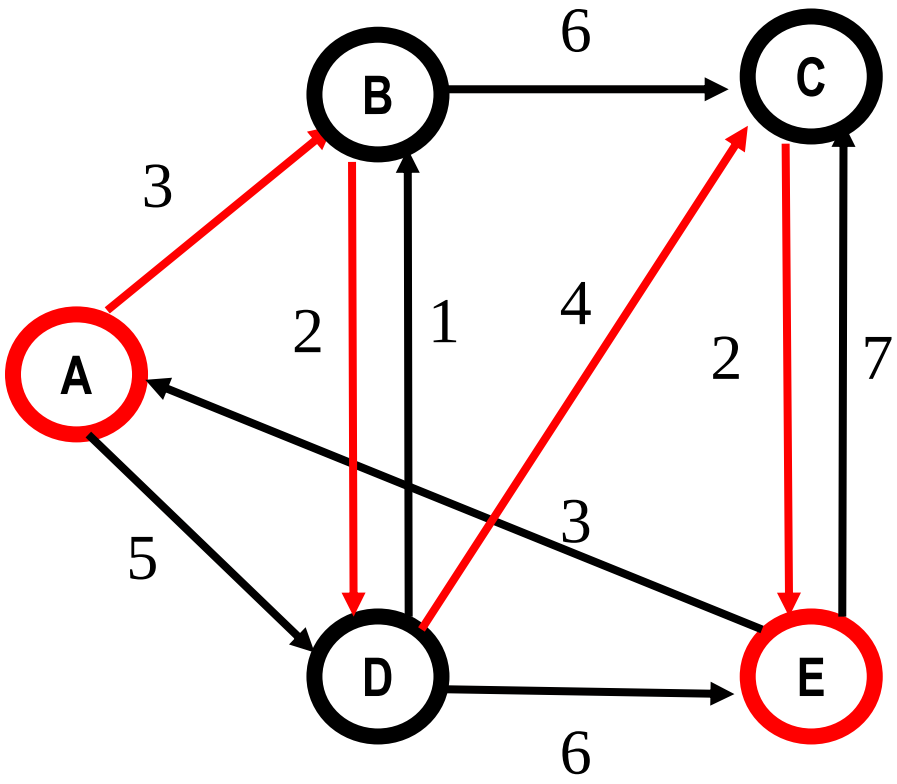


Example: A Shortest-Path from A to E



P1 = A, D, E 11

Example: Another Shortest-Path from A to E

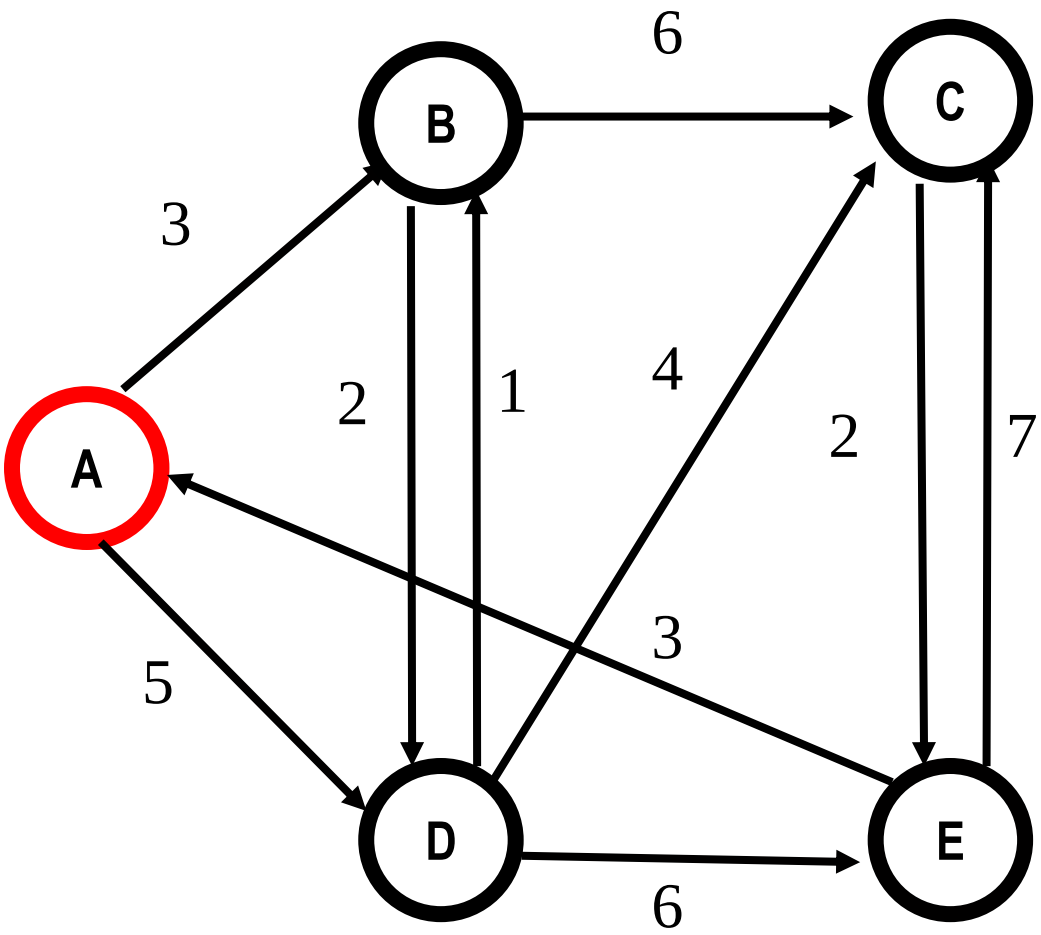


P2 = A, B, D, C, E 11

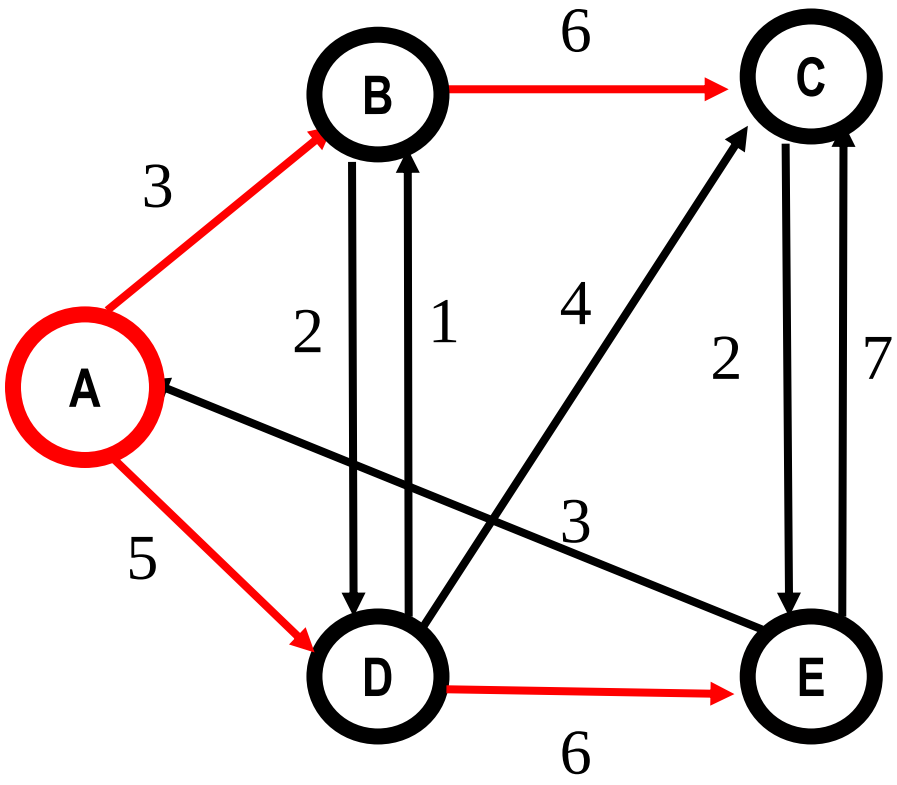
2. The Single-Source Shortest Paths Problem

- Given a weighted, directed graph G=(V,E) with weight function W: E $\rightarrow$ R mapping edges to real valued weights,
- Find a shortest path from a given source vertex s to every vertex v in V.
  - ➔ The Single-Source Shortest Paths Problem
  - ➔ The Single-Source/All-Destinations Shortest Paths Problem

Example: Shortest-Paths from A?



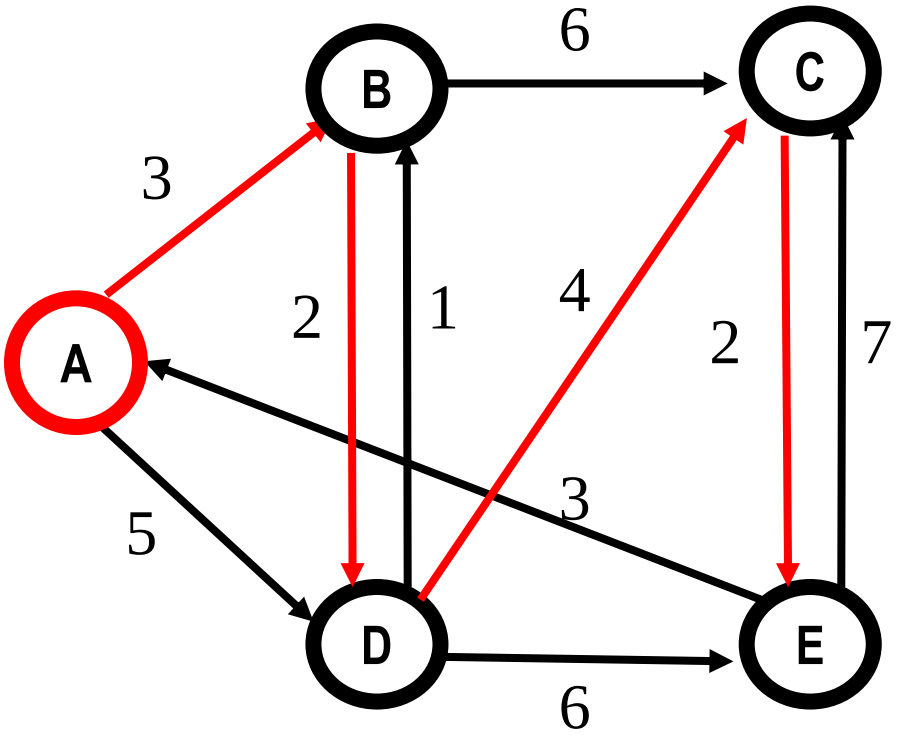
Example: Shortest-Path from A



A

A	B	C	D	E
0	3	9	5	11

Example: Shortest-Path from A



A

A	B	C	D	E
0	3	9	5	11

The Single-Source Shortest Paths Problem

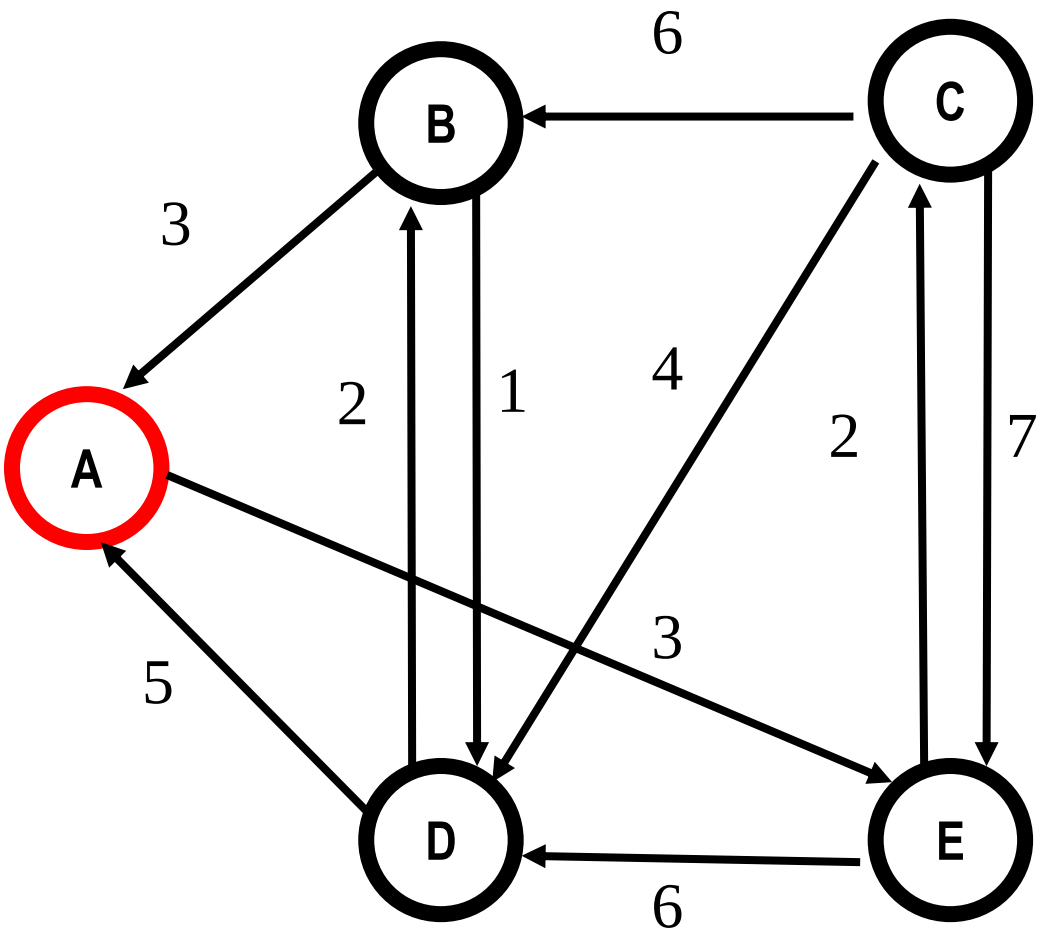
- The **brute-force** algorithm is to consider all possible factorial paths and take the shortest.

- **Bellman-Ford Algorithm**
  - ➔**Dynamic Programming**
- **Dijkstra Algorithm**
  - ➔**Greedy Approach**

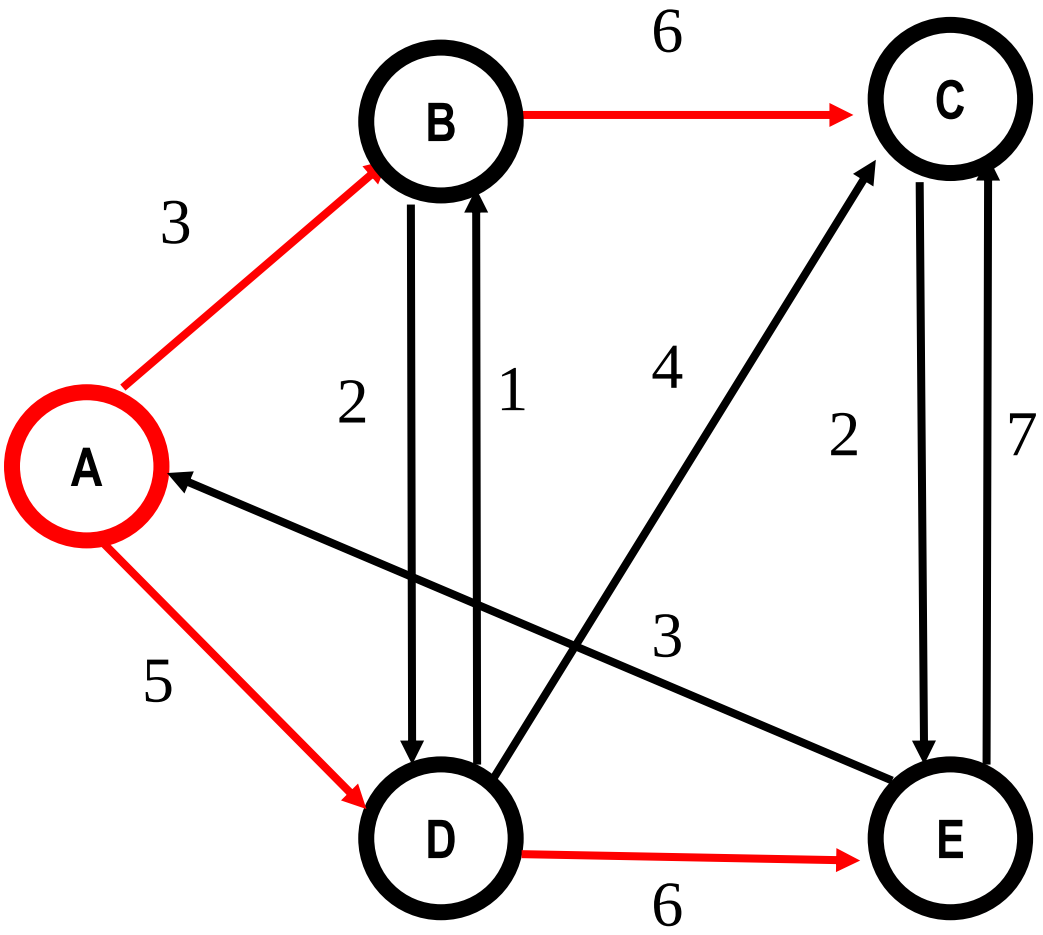
3. The Single-Destination Shortest Paths Problem

- Given a weighted, directed graph G=(V,E) with weight function W: E → R mapping edges to real valued weights,
- Find a shortest path to a given destination vertex d from every vertex v in V.
  - ➔ The Single-Destination Shortest Paths Problem
  - ➔ The Single-Destination/All-Sources Shortest Paths Problem
    - ☞ The reverse of the single-source shortest-paths problem!

Example: Shortest-Paths to A



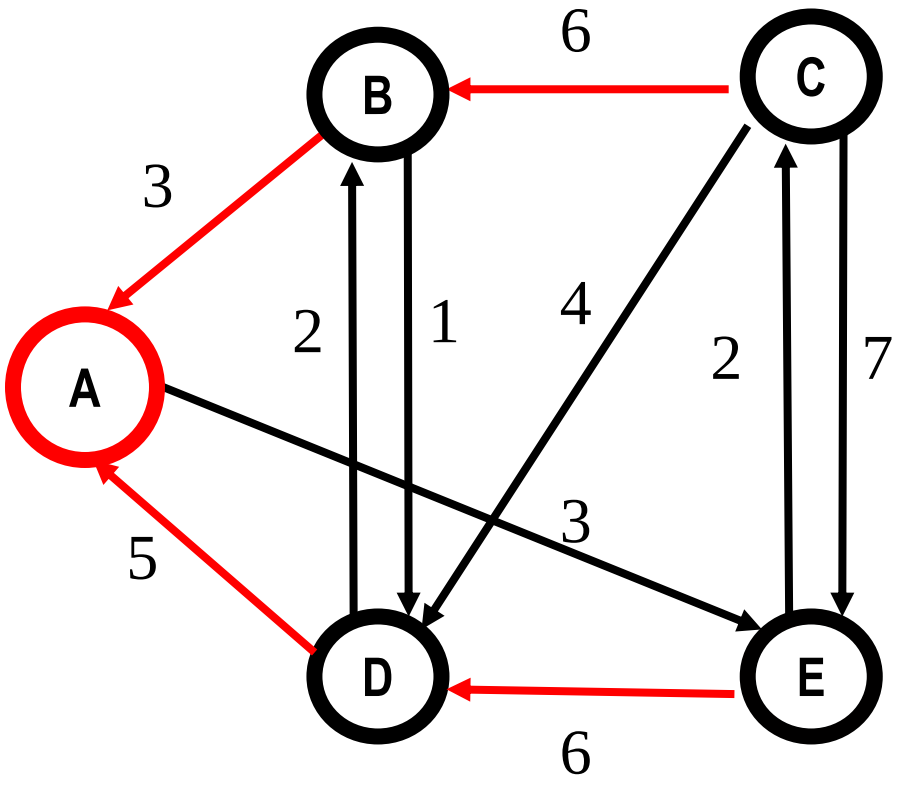
Example: Shortest-Paths from A



A

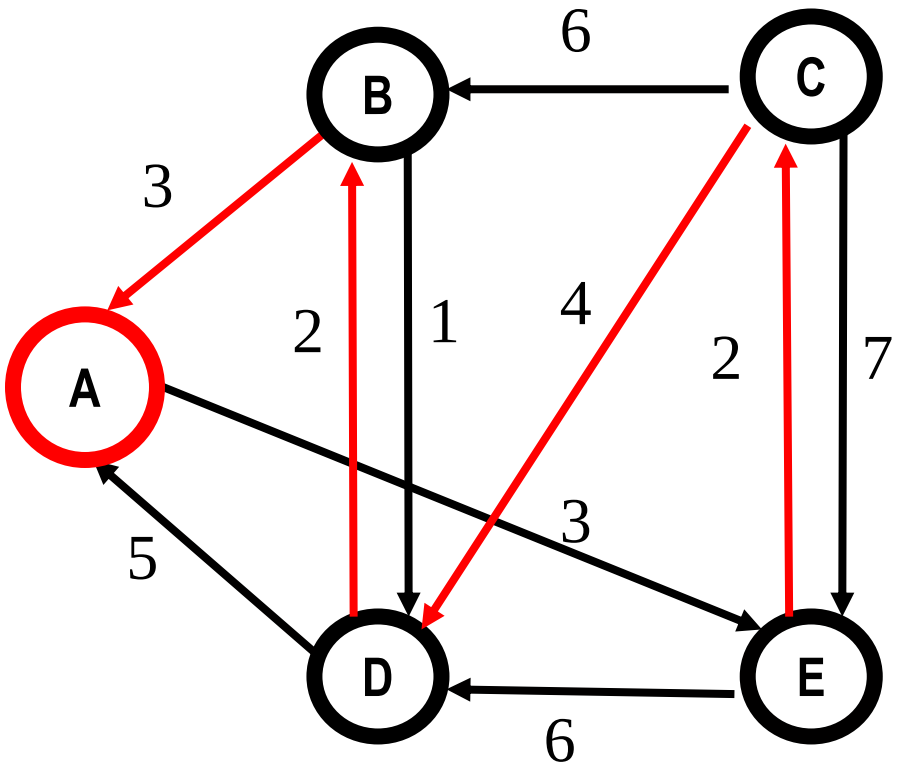
A	B	C	D	E
0	3	9	5	11

Example: Shortest-Paths to A



A	B	C	D	E
0	3	9	5	11

Example: Shortest-Paths to A

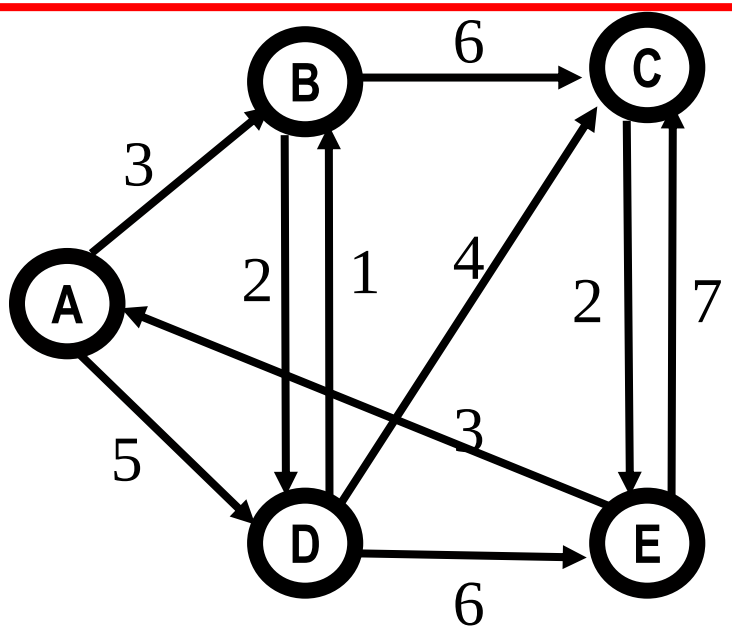


A	B	C	D	E
0	3	9	5	11

4. The All-Pairs Shortest Paths Problem

- Given a weighted, directed graph G=(V,E) with weight function W: E $\rightarrow$ R mapping edges to real valued weights,
- Find a shortest path from u to v for every pair of vertices u and v in V.
  - ➔ The All-Pairs Shortest Paths Problem

Example: Shortest-paths for Every Pair of Vertices



A

A	B	C	D	E
0	3	9	5	11

B

A	B	C	D	E

C

A	B	C	D	E

D

A	B	C	D	E

F

A	B	C	D	E

The All-Pairs Shortest Paths Problem

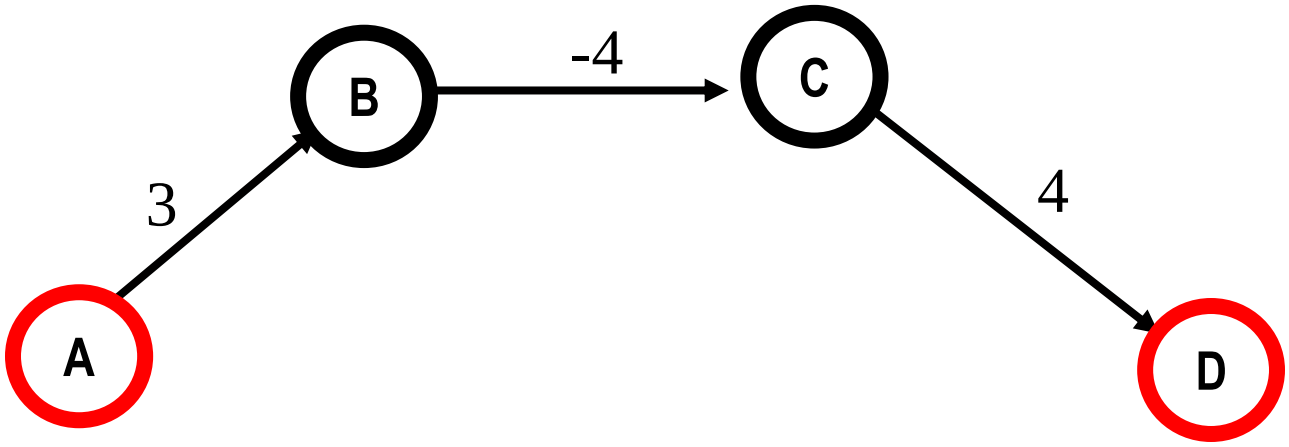
- The **brute-force** algorithm is to consider all possible factorial paths and take the shortest.

- **Floyd-Warshall Algorithm**
  - ➔**Dynamic Programming**

Negative-Weight Edges? In The Shortest-Paths Problems

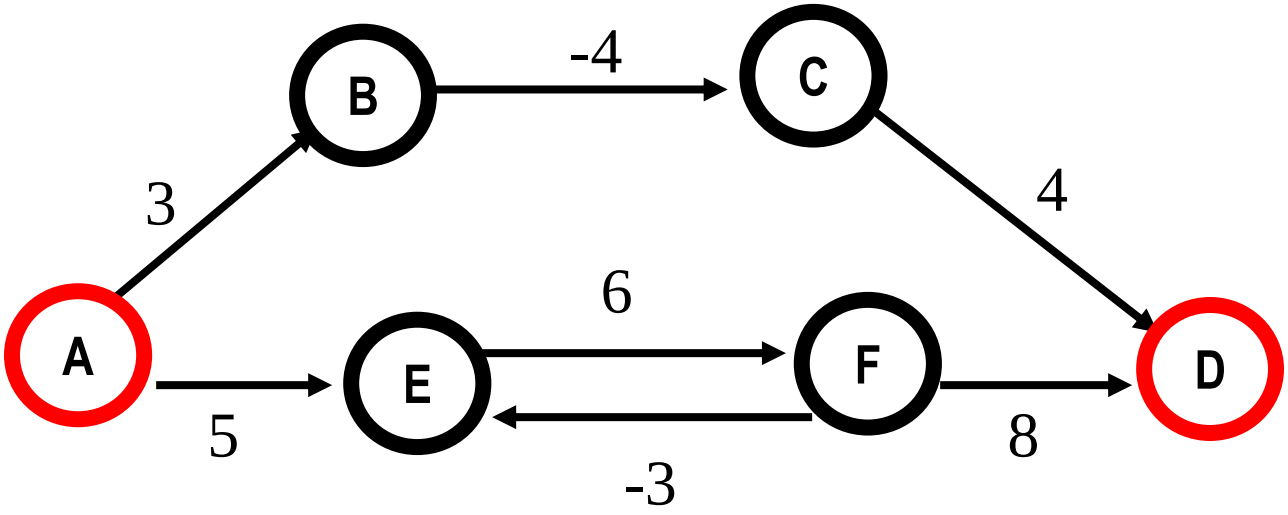
- Any problem with edges with **negative weights**?
  - ➔ Yes and No!

Example: Negative-Weight Edges?



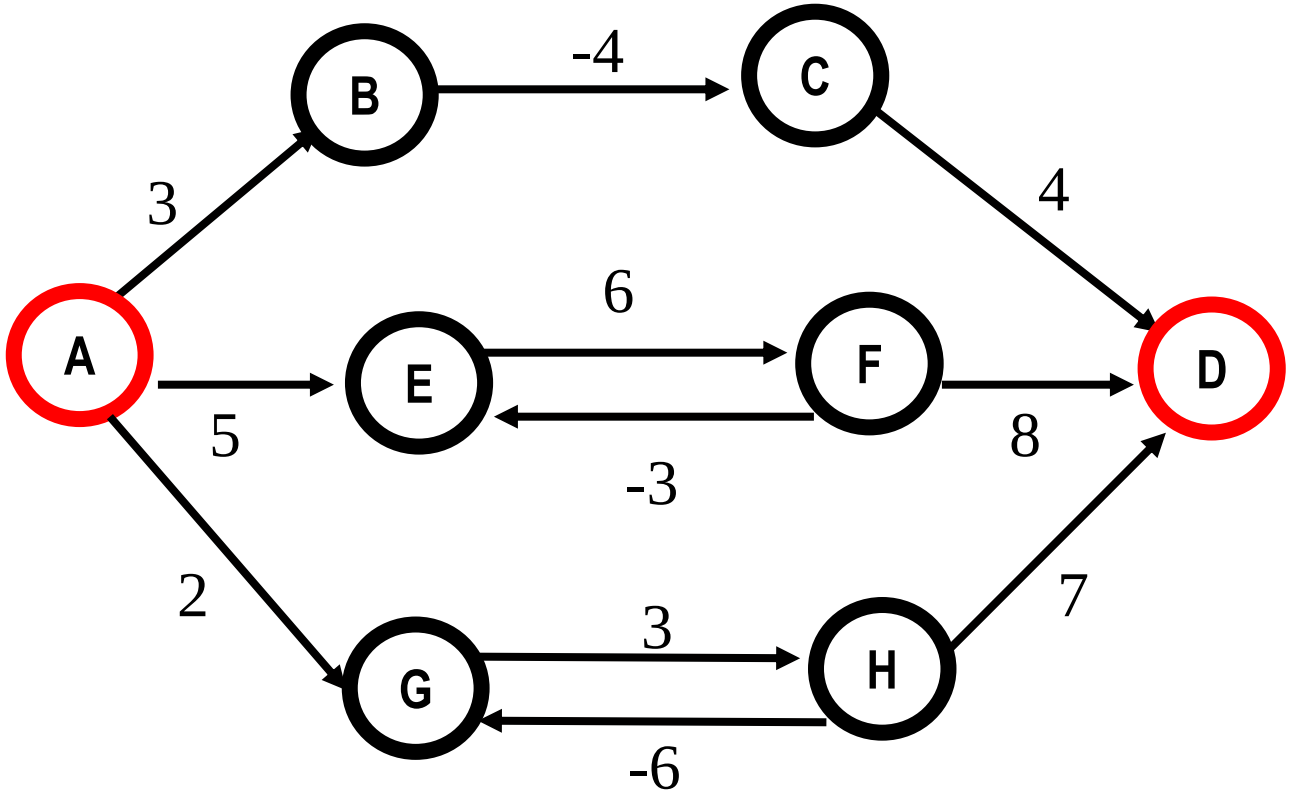
A shortest path from A to D ?

Example: Negative-Weight Edges + Nonnegative Cycles?



A shortest path from A to D ?

Example: Negative-Weight Edges + Negative Cycles?



A shortest path from A to D ?

The Shortest-Paths Problem with Negative Edges & Negative Cycles

- Is the shortest path problem defined for weighted directed graphs that contain negative edges but there is no negative cycle?
  - ➔ Yes
- Is the shortest path problem defined for weighted directed graphs that contain negative cost cycles?
  - ➔ No
- The shortest-path problem is greatly simplified when all edges have non-negative weights!

►QUIZ?

- Types of Shortest Path Problems?
  - ➔SPSP
  - ➔SSSP
  - ➔SDSP
  - ➔APSP
- **TSP (Traveling Salesperson Problem)?**

Homework Assignment

►Homework Assignment?

- Compare **depth-first-search** with **breadth-first-search** of graphs.

the right majors, minors &

ucation?

or students' academic and career success

ging for many students - *Many students change their during college!*

e prediction of student success in MMC could dual students

d their right MMC

chieve their academic goals

END

Prof. Young Park