COSC 2436: Graphs

Introduction, Depth-First Search, Breadth-First Search

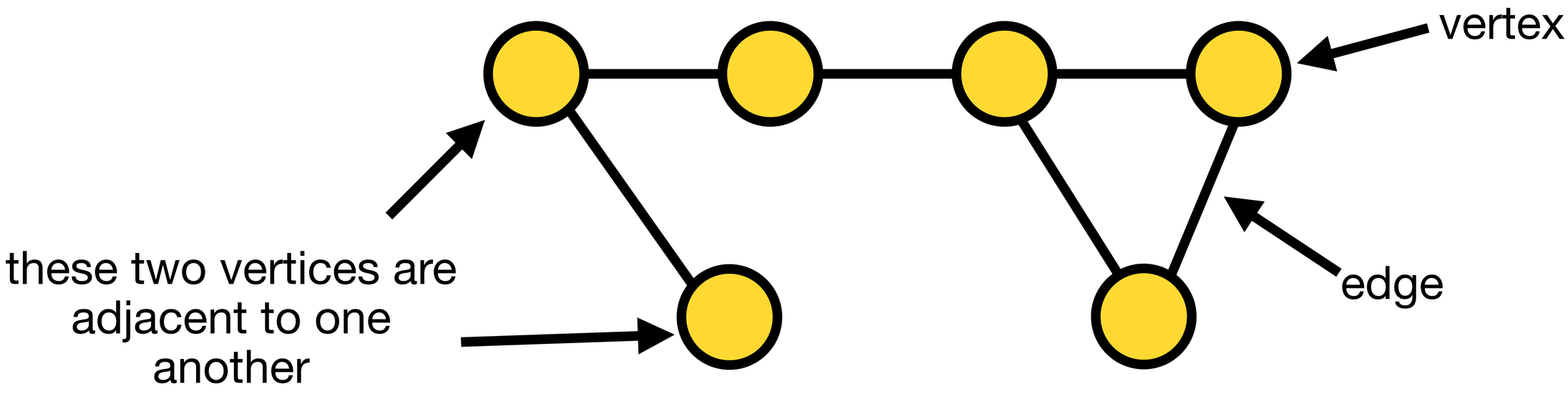
J. Eoin Donovan - 2023

What is a graph?

- **A graph is a data strucutre used to represent relationships and connections between entities**

Graph terms to know

- **vertex - represents an entity (node)**
- **edge - represents a connection**
- **adjacent - there is an edge (connection) between two vertices**

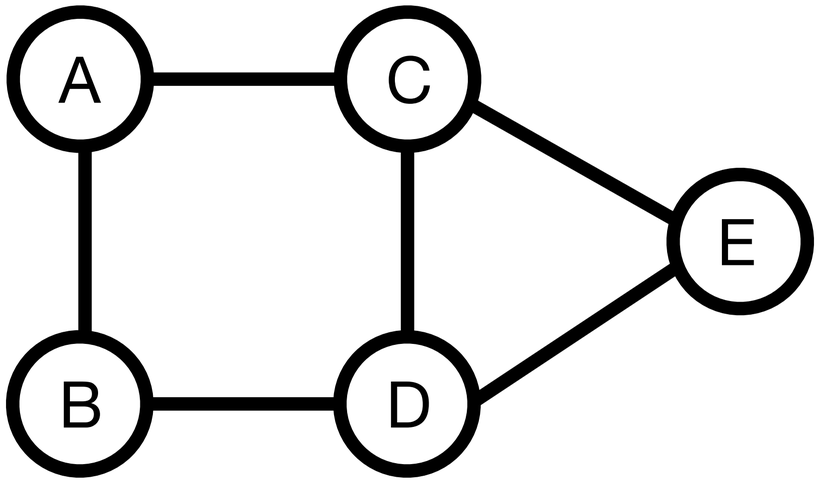


Different types of graphs

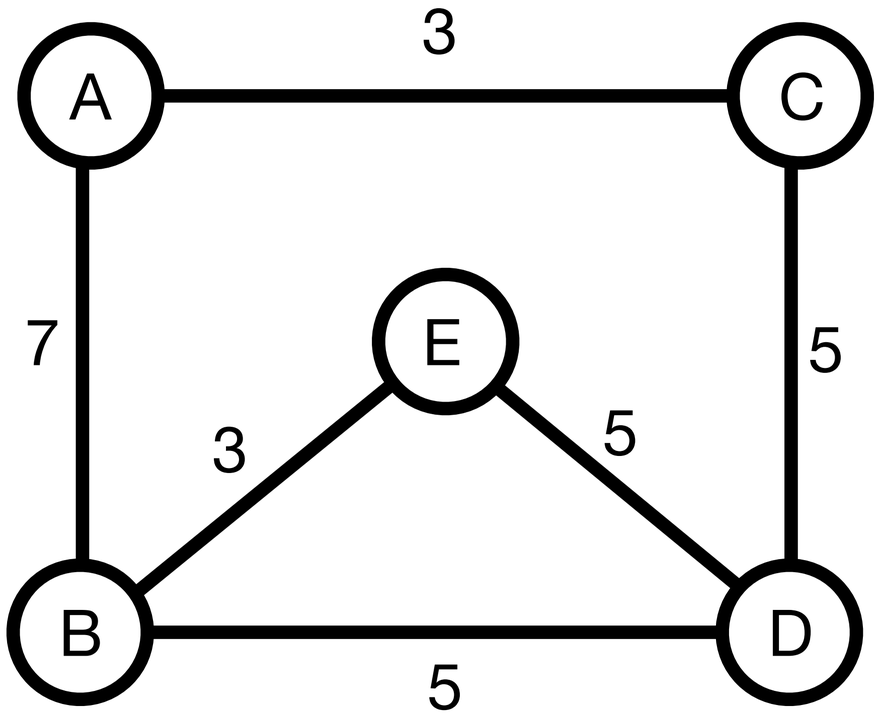
The edges of graphs can either be:

- **weighted/unweighted**
- **directed/undirected**

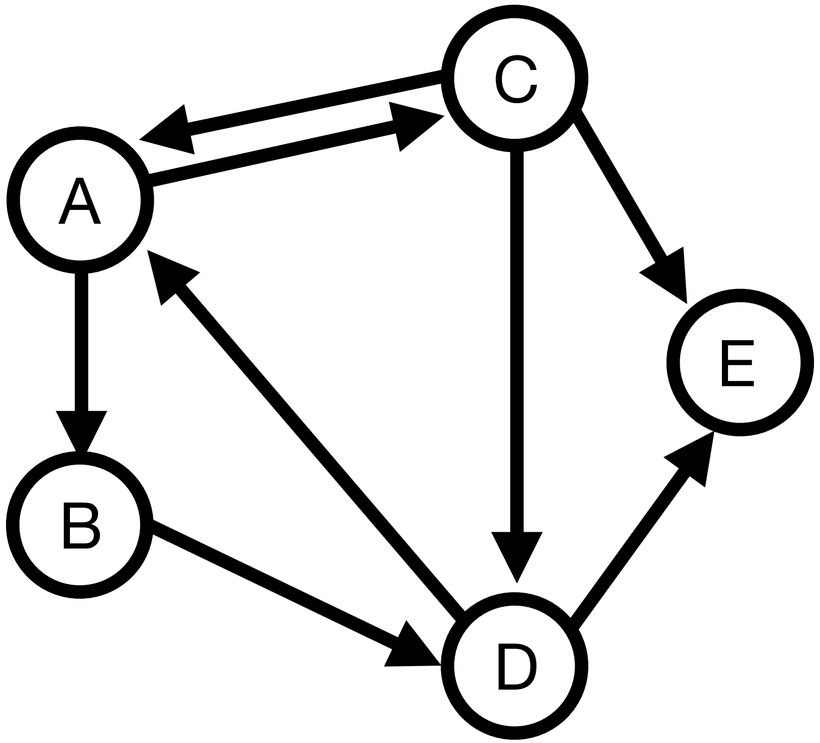
Unweighted, Undirected Graph



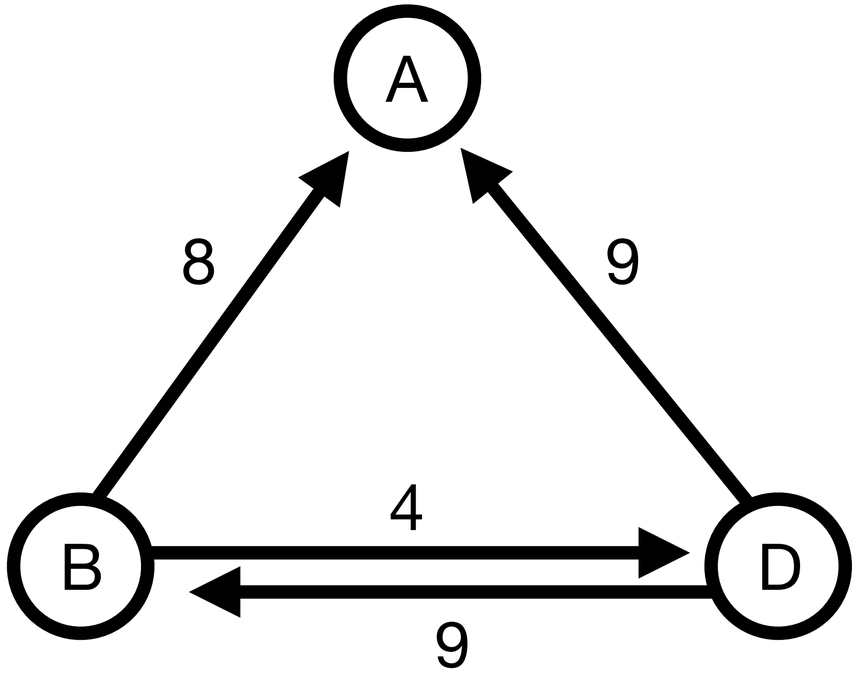
Weighted, Undirected Graph



Unweighted, Directed Graph



Weighted, Directed Graph

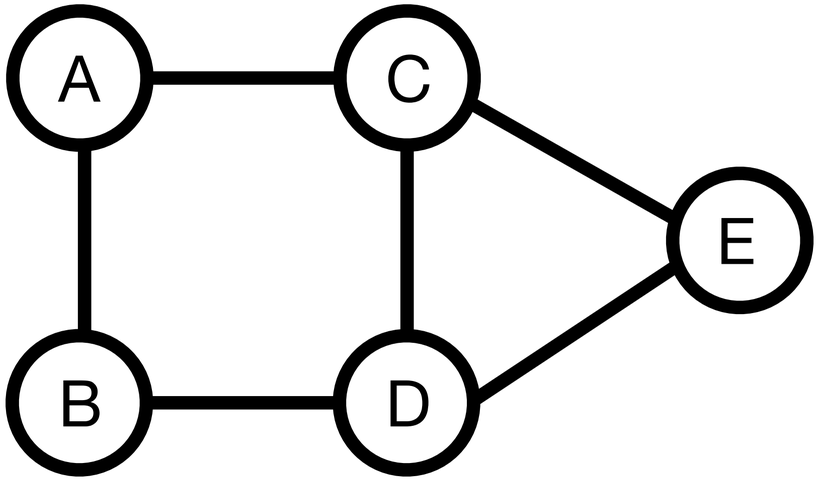


How to represent a graph

There are two main ways to represent a graph:

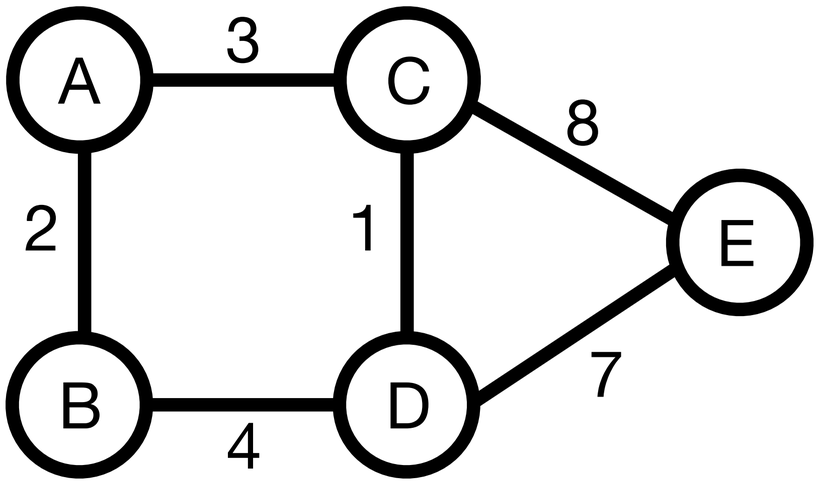
- **Adjacency Matrix**
- **Adjacency List**

Graph as Adjacency Matrix



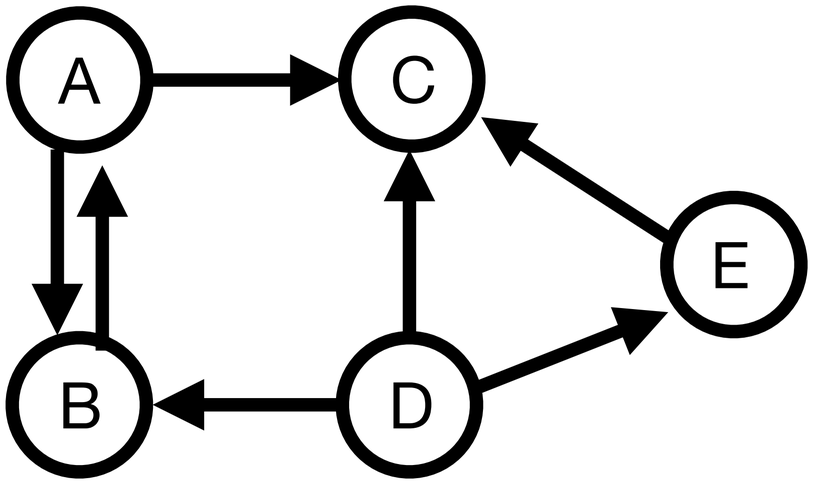
	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	1	0
C	1	0	0	1	1
D	0	1	1	0	1
E	0	0	1	1	0

Graph as Adjacency Matrix



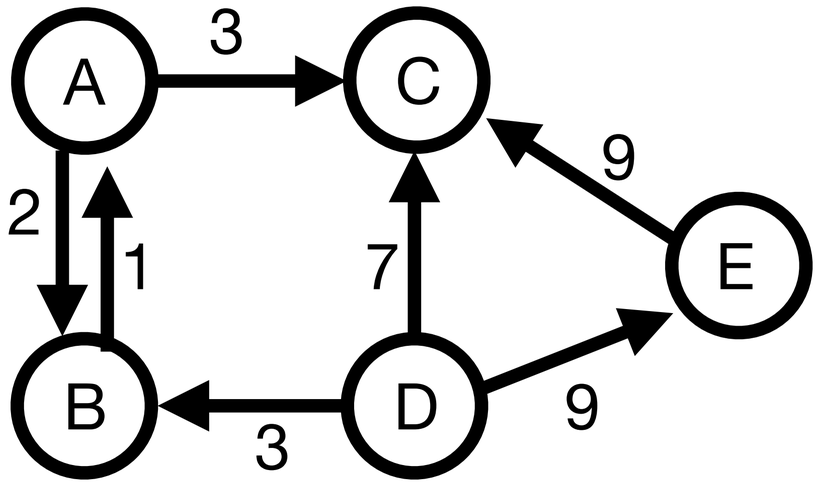
	A	B	C	D	E
A	0	2	3	0	0
B	2	0	0	4	0
C	3	0	0	1	8
D	0	4	1	0	7
E	0	0	8	7	0

Graph as Adjacency Matrix



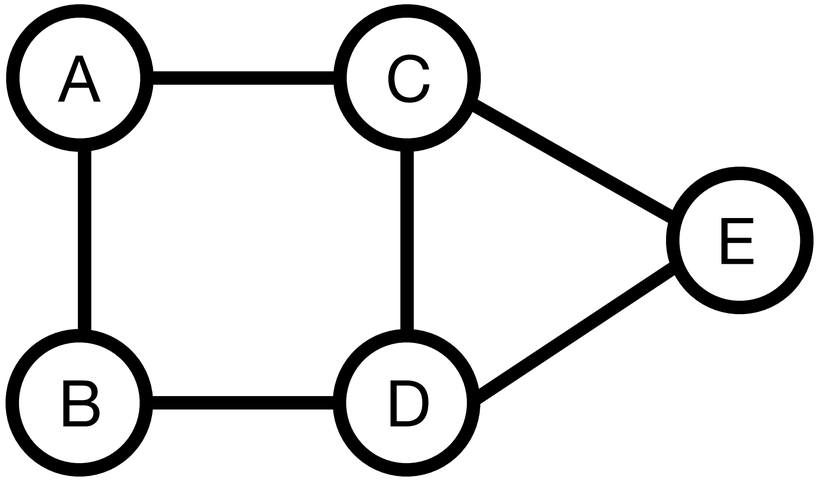
	A	B	C	D	E
A	0	1	1	0	0
B	1	0	0	0	0
C	0	0	0	0	0
D	0	1	1	0	1
E	0	0	1	0	0

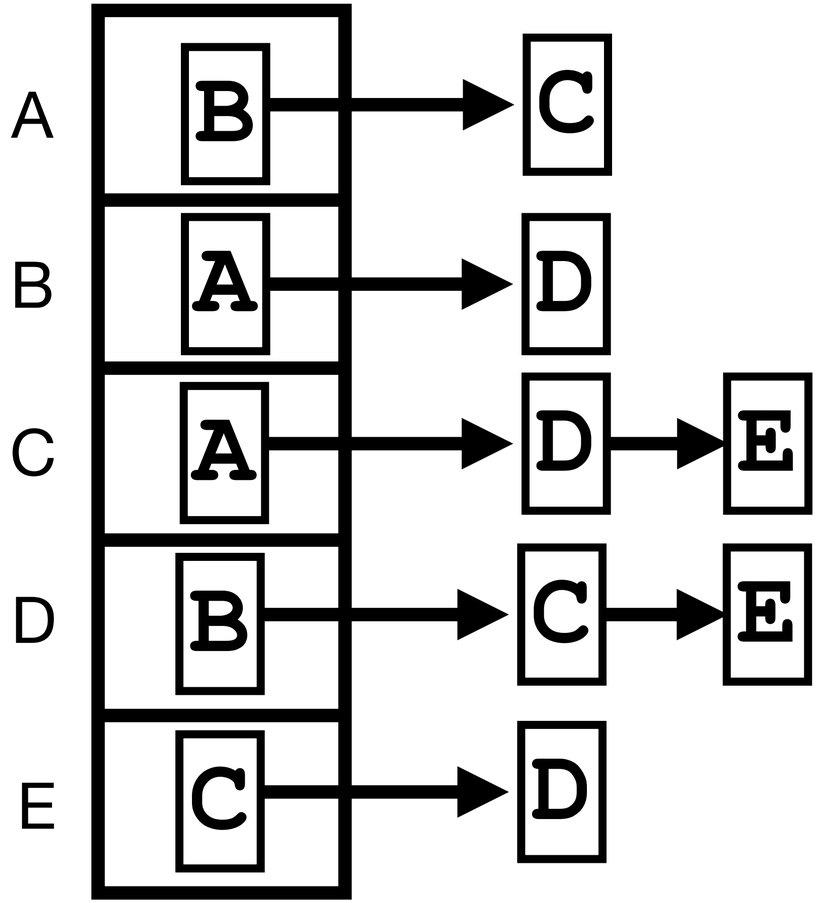
Graph as Adjacency Matrix



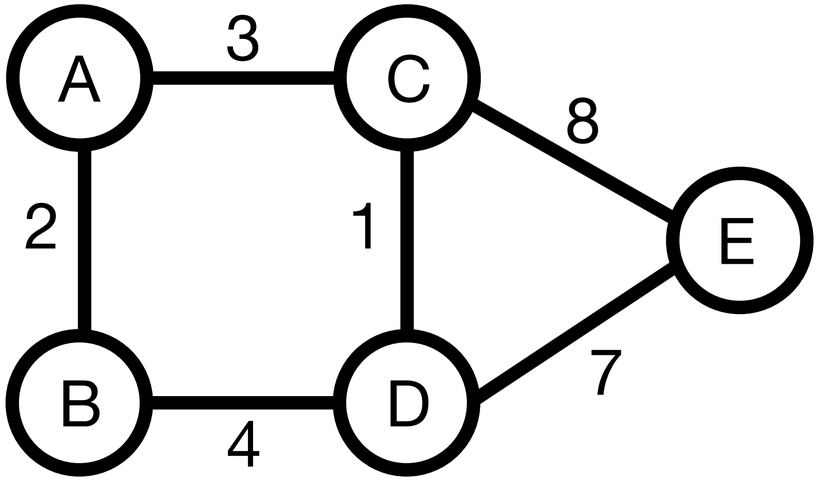
	A	B	C	D	E
A	0	2	3	0	0
B	1	0	0	0	0
C	0	0	0	0	0
D	0	3	7	0	9
E	0	0	9	0	0

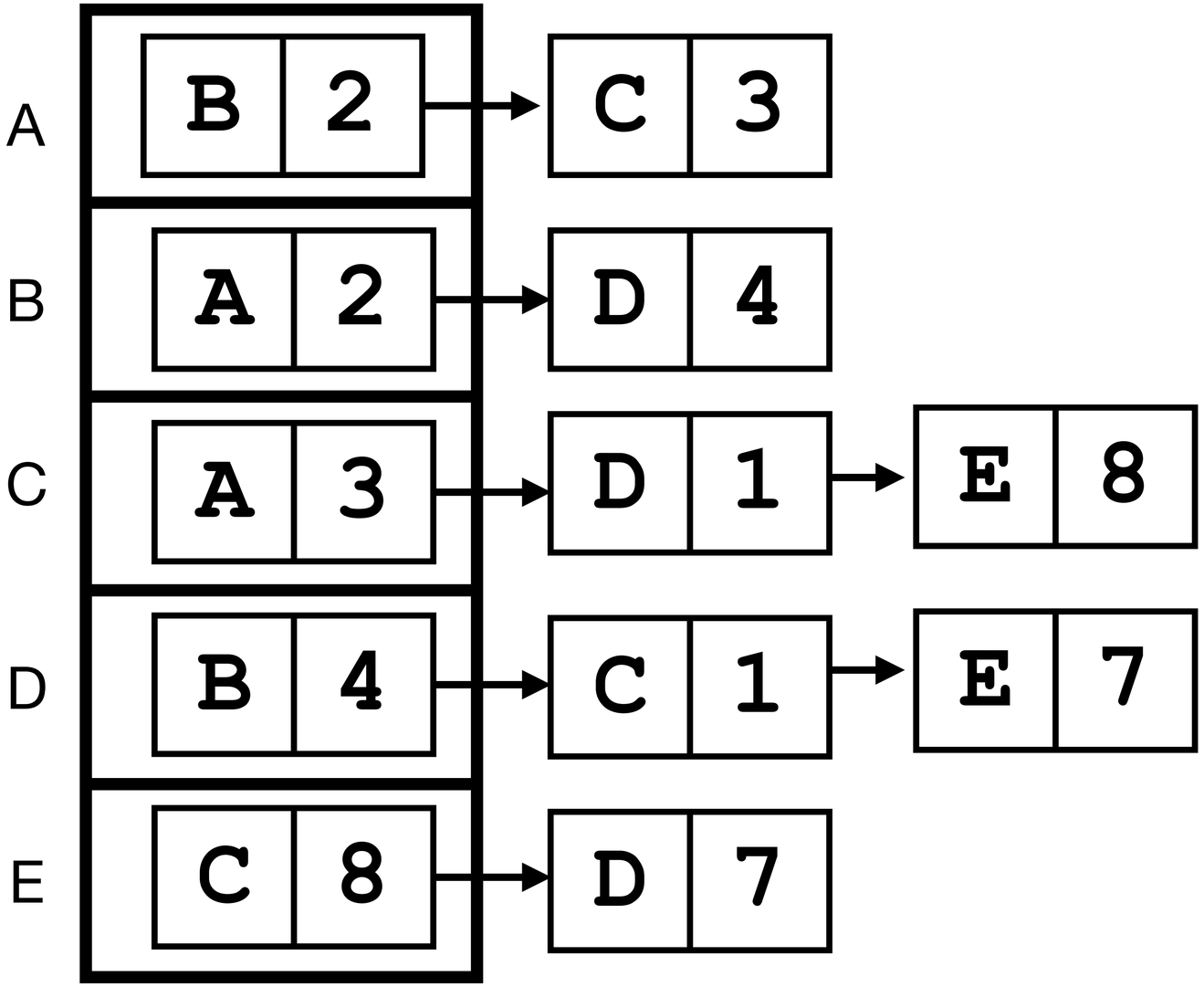
Graph as Adjacency List



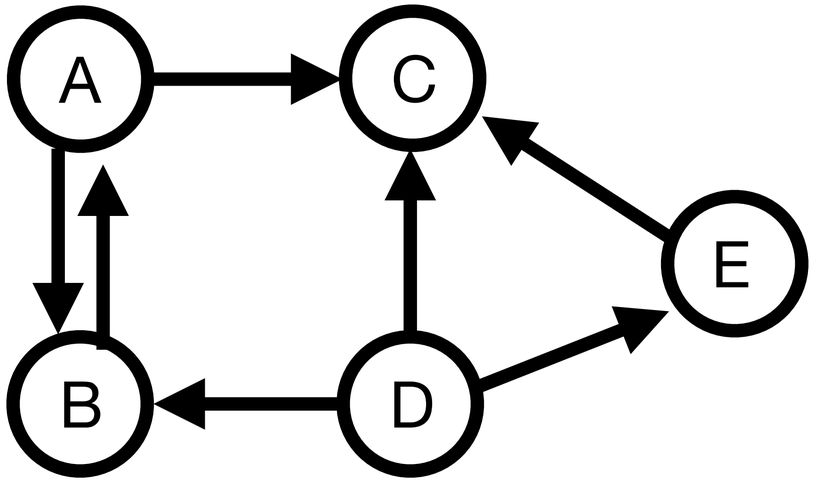


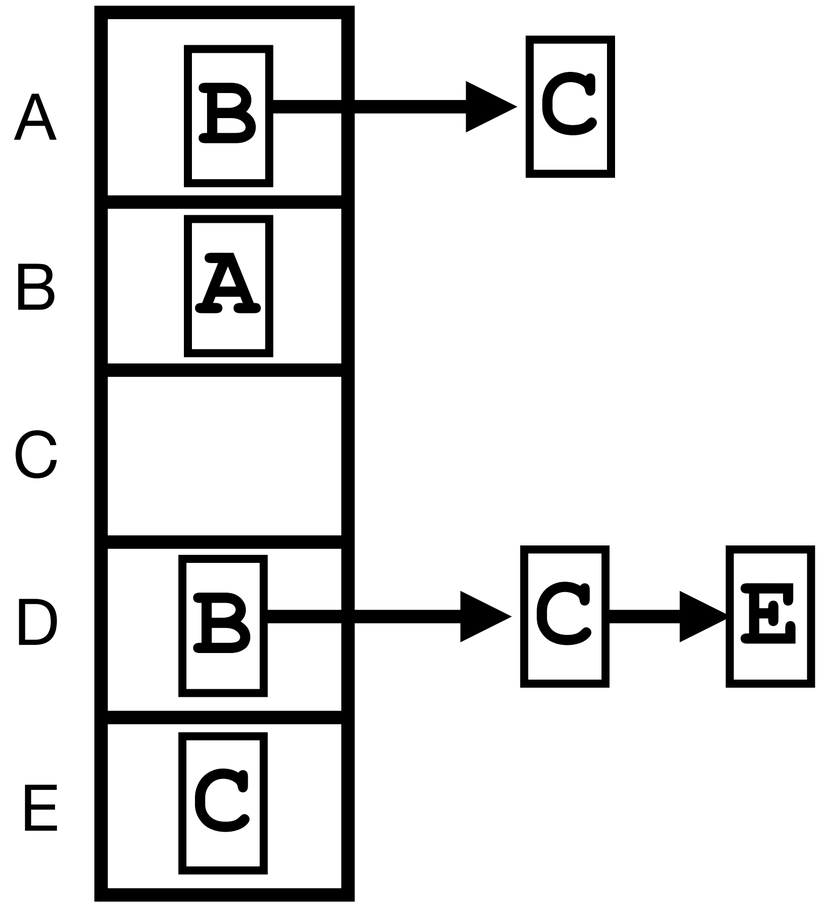
Graph as Adjacency List



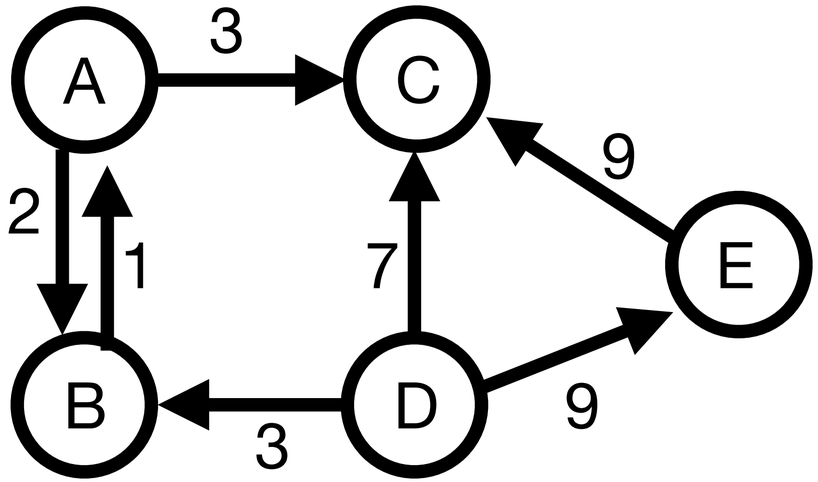


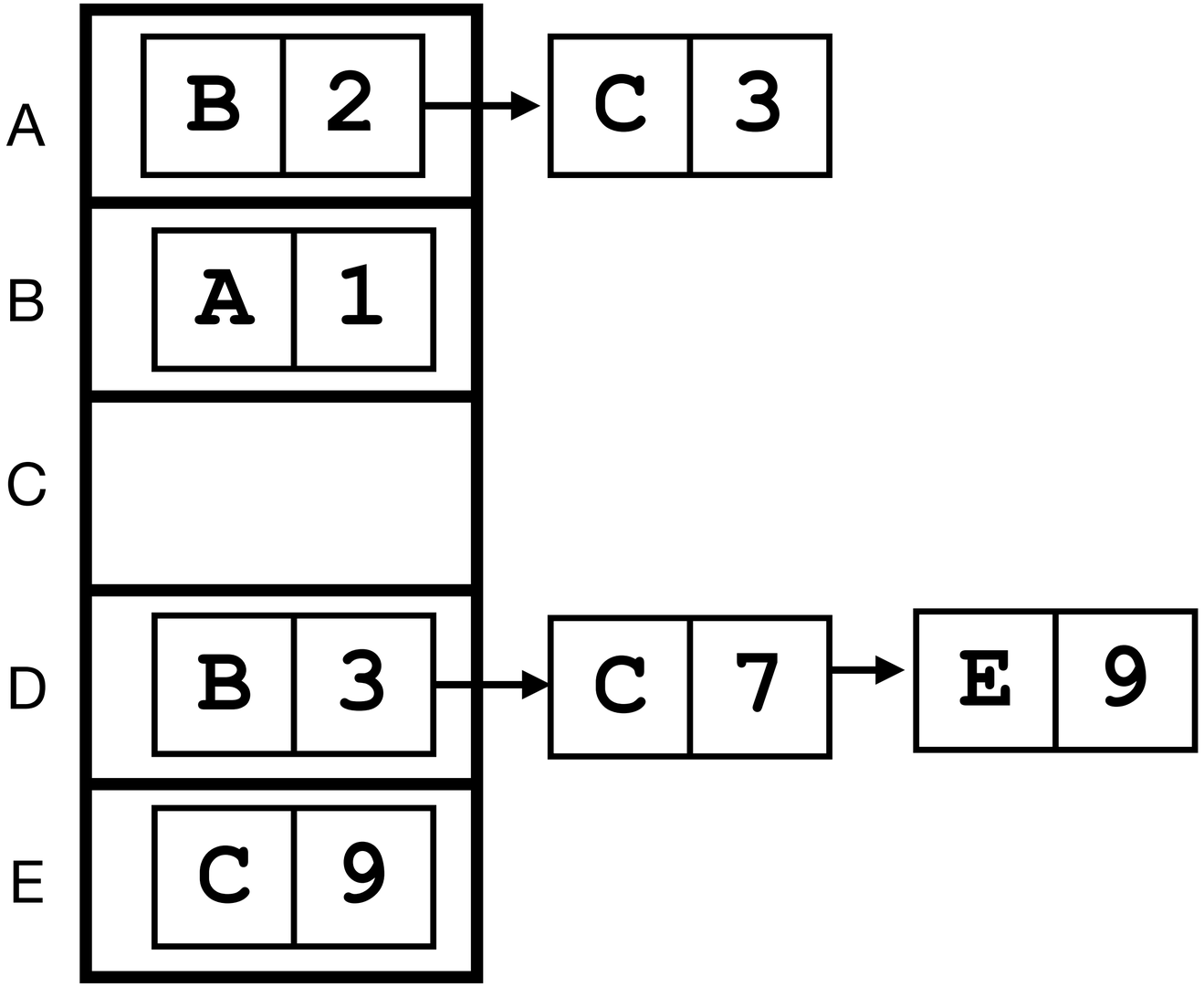
Graph as Adjacency List





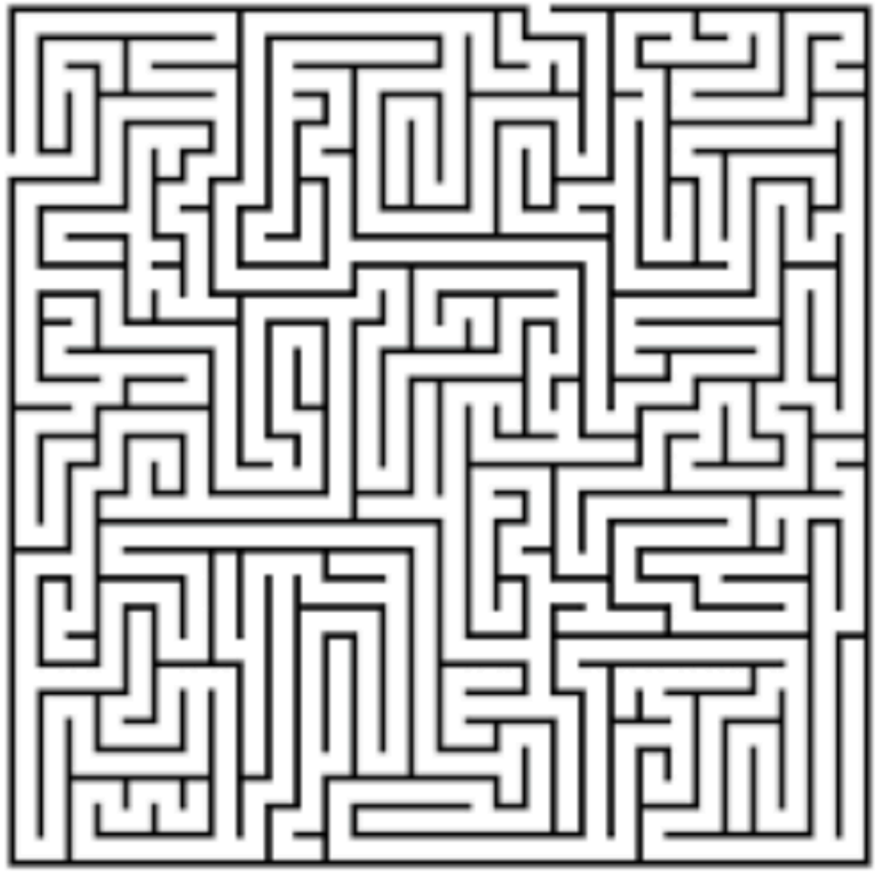
Graph as Adjacency List





Depth First Search (DFS)

- DFS is a graph traversal algorithm
- DFS explores a graph by moving as deep as possible along each “path” before backtracking
- DFS can be implemented recursively or iteratively using a stack



DFS Algorithm

```
for each vertex, v, in the graph
   if v is not visited
     start the depth first traversal at v
```

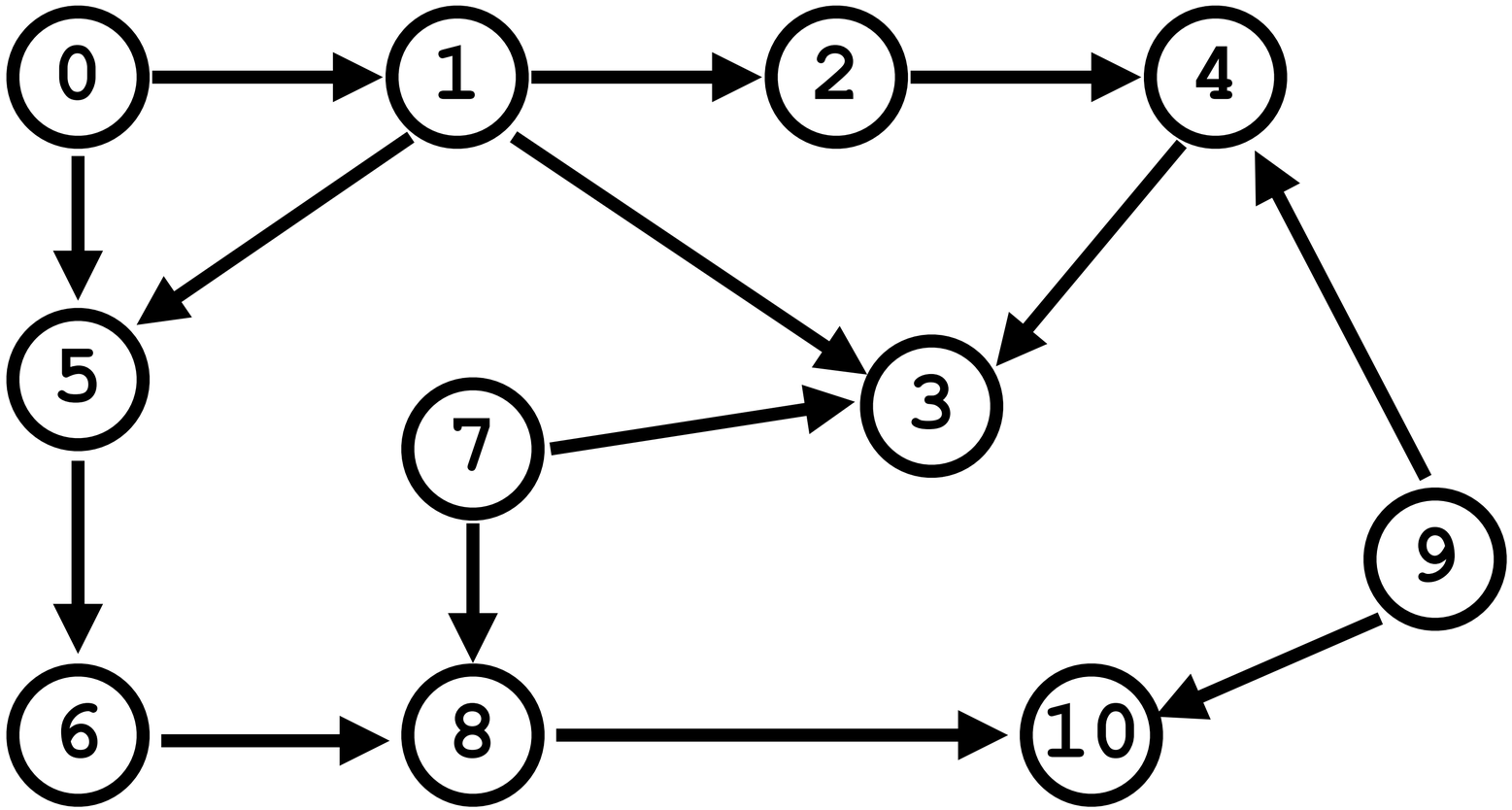
Depth-First Traversal at a given node, `v`

1. `mark node v as visited`
2. `visit the node`
3. `for each vertex u adjacent to v`

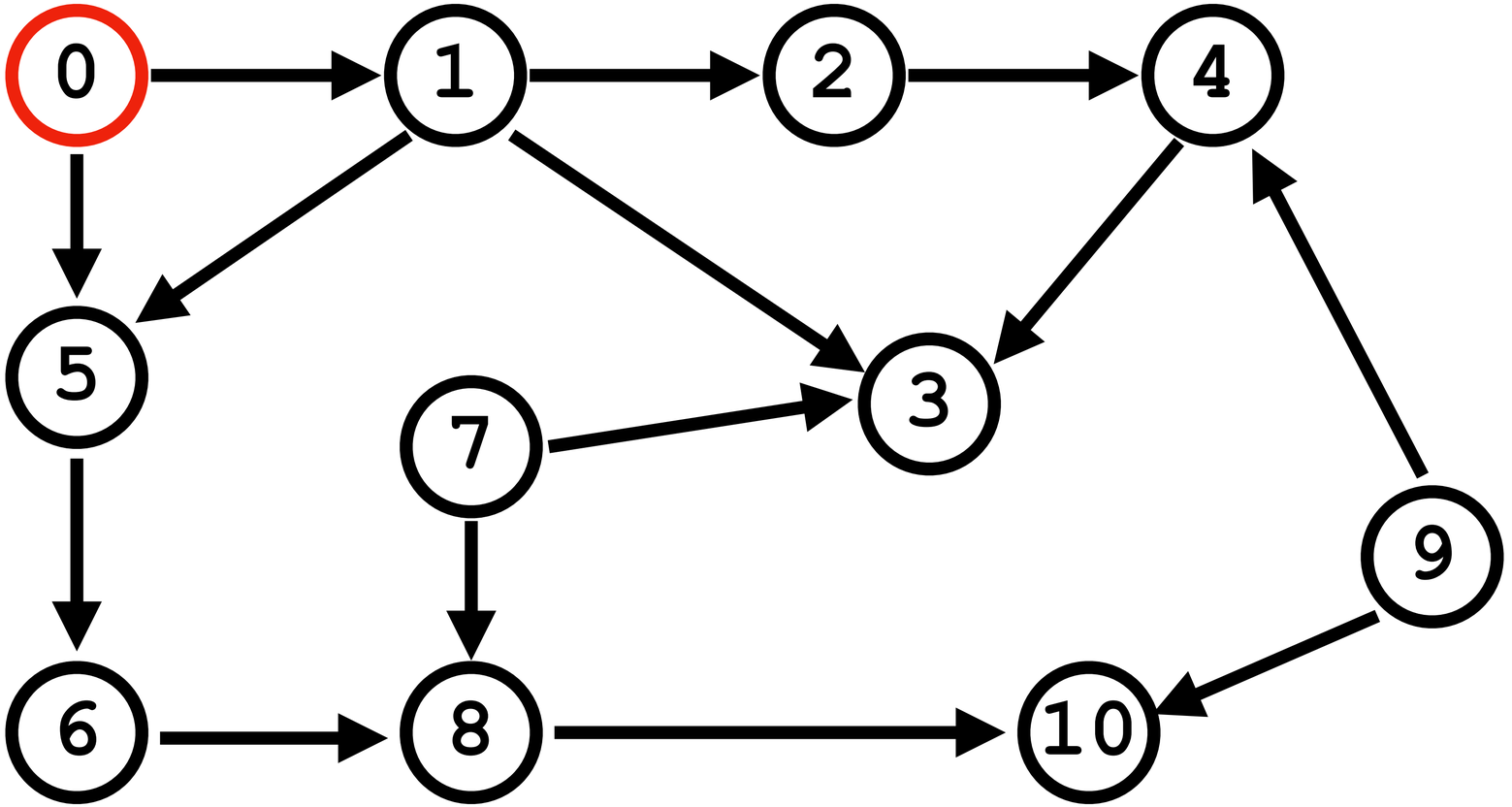
```
        if u us not visited
           start the depth first traversal at u
```

DFS Tracing

Let’s trace the depth first search of the graph below staring from veretx 0

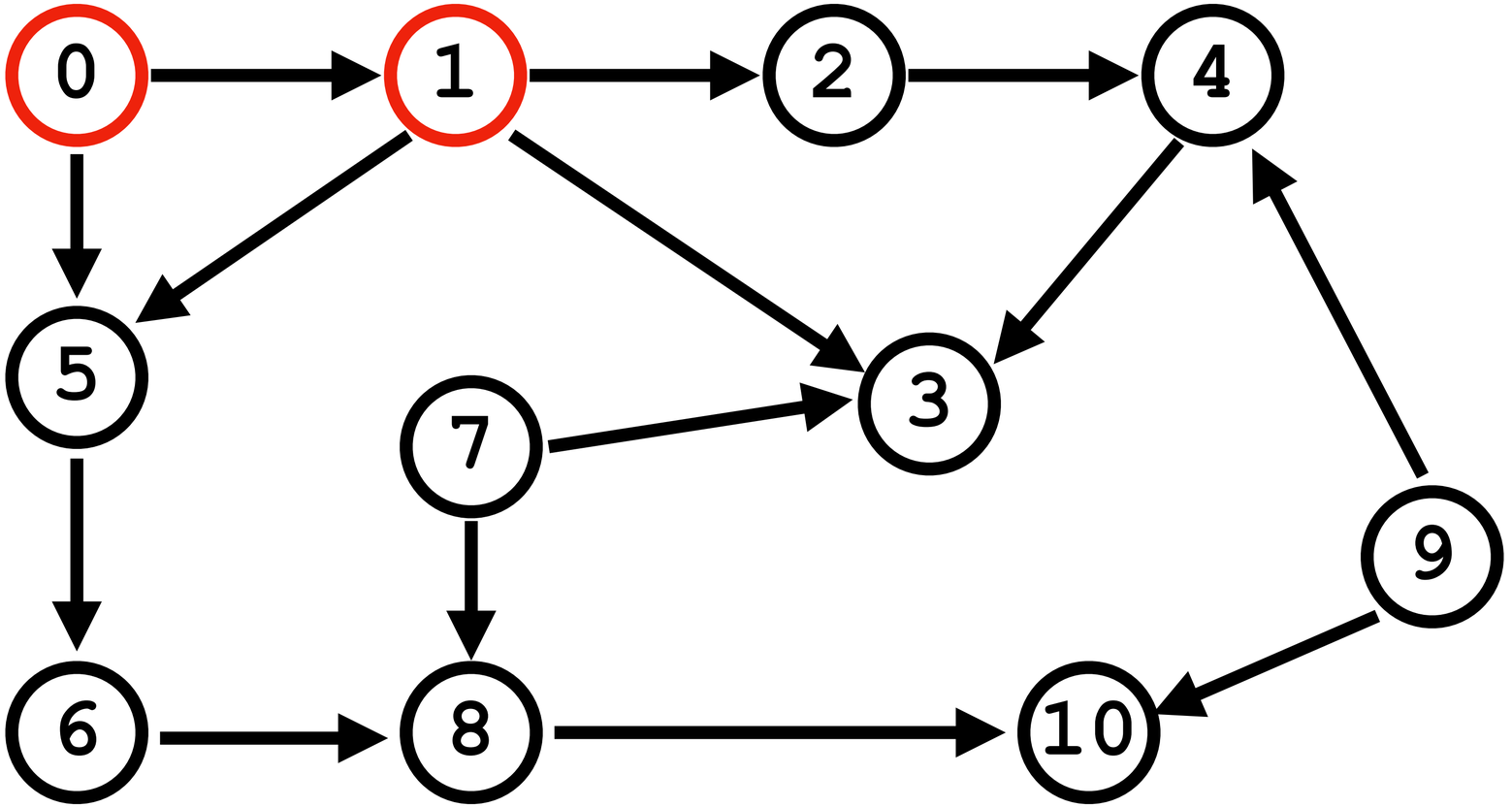


DFS Tracing



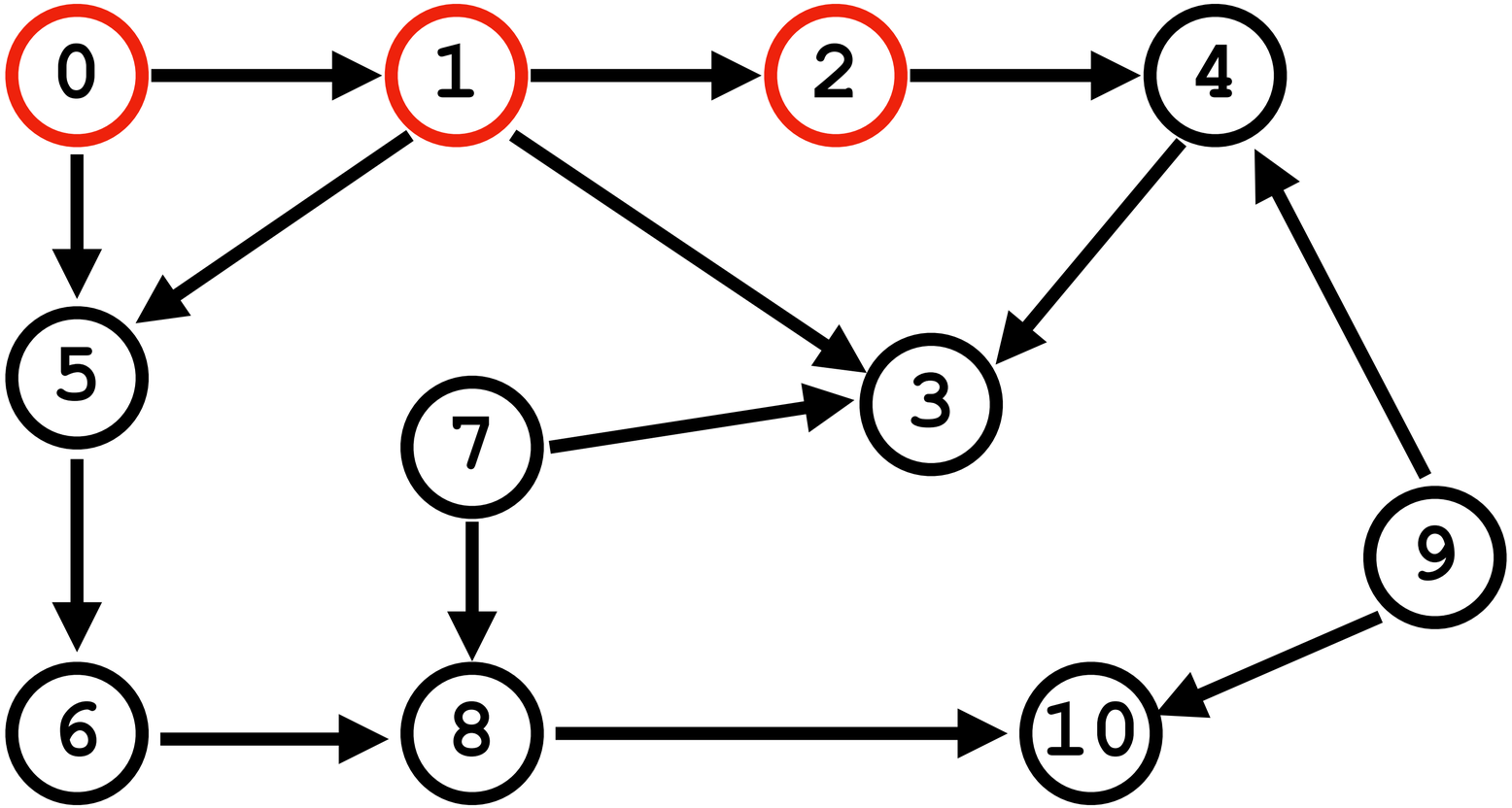
0

DFS Tracing



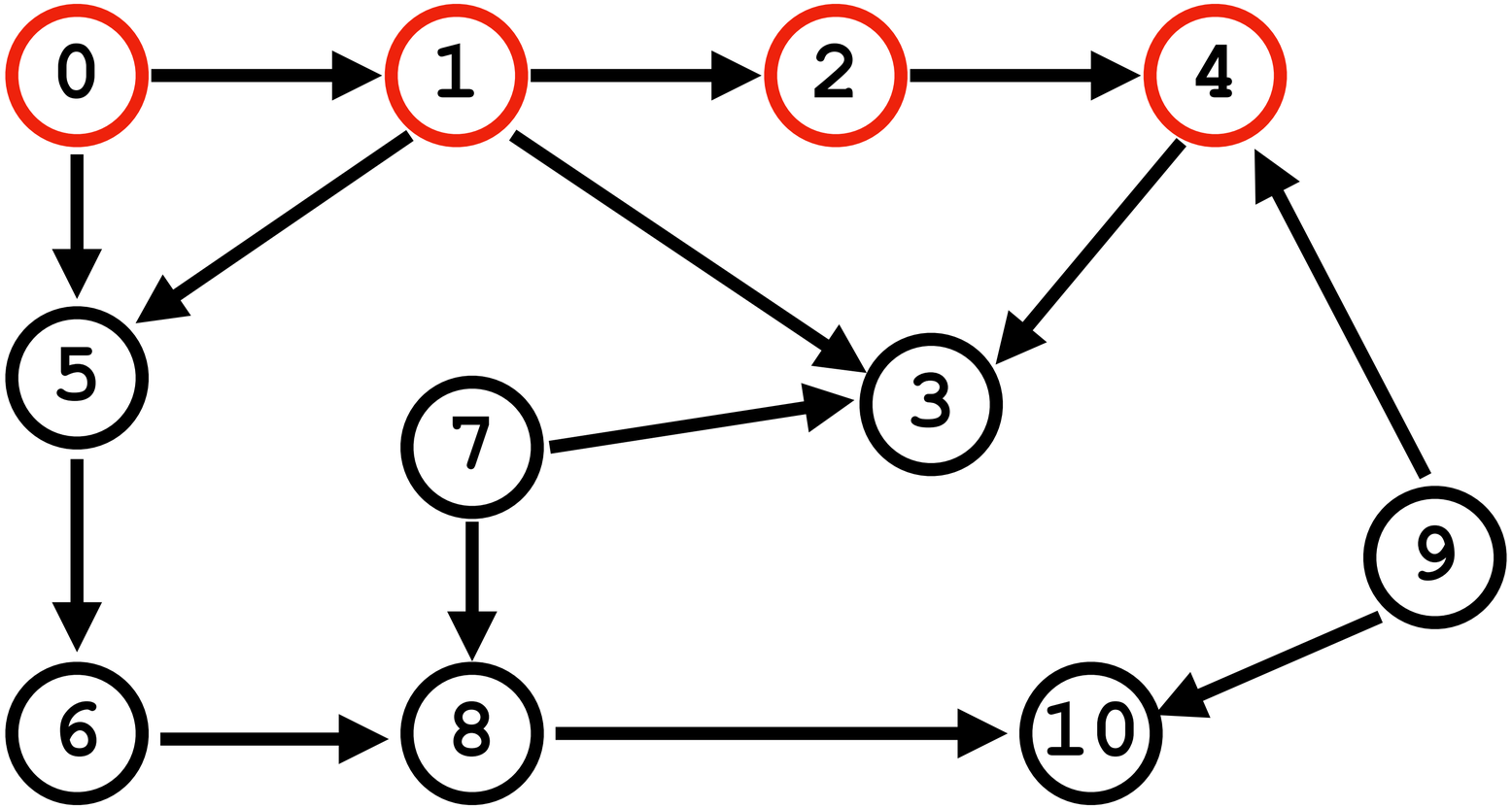
0 1

DFS Tracing



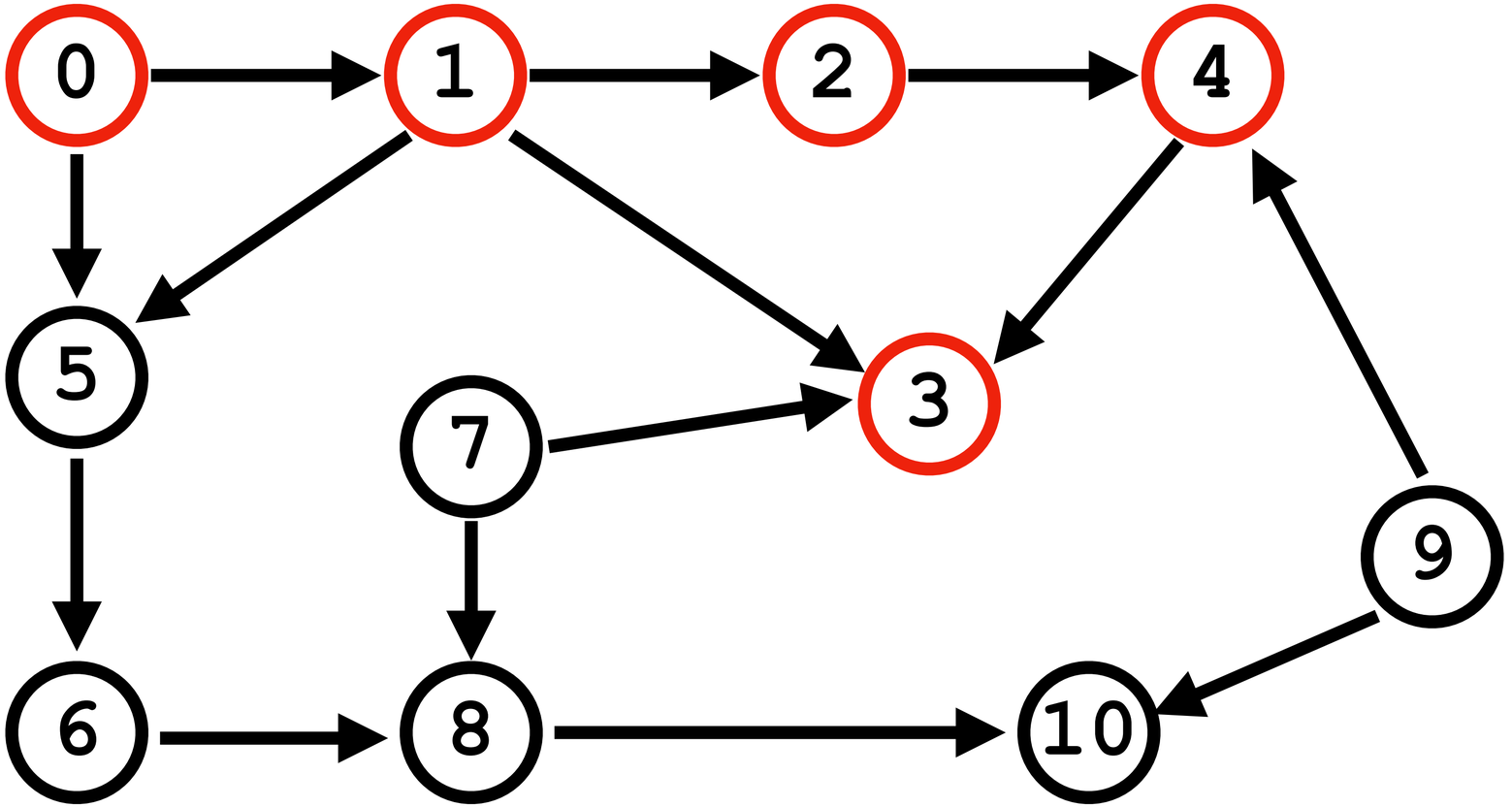
0 1 2

DFS Tracing



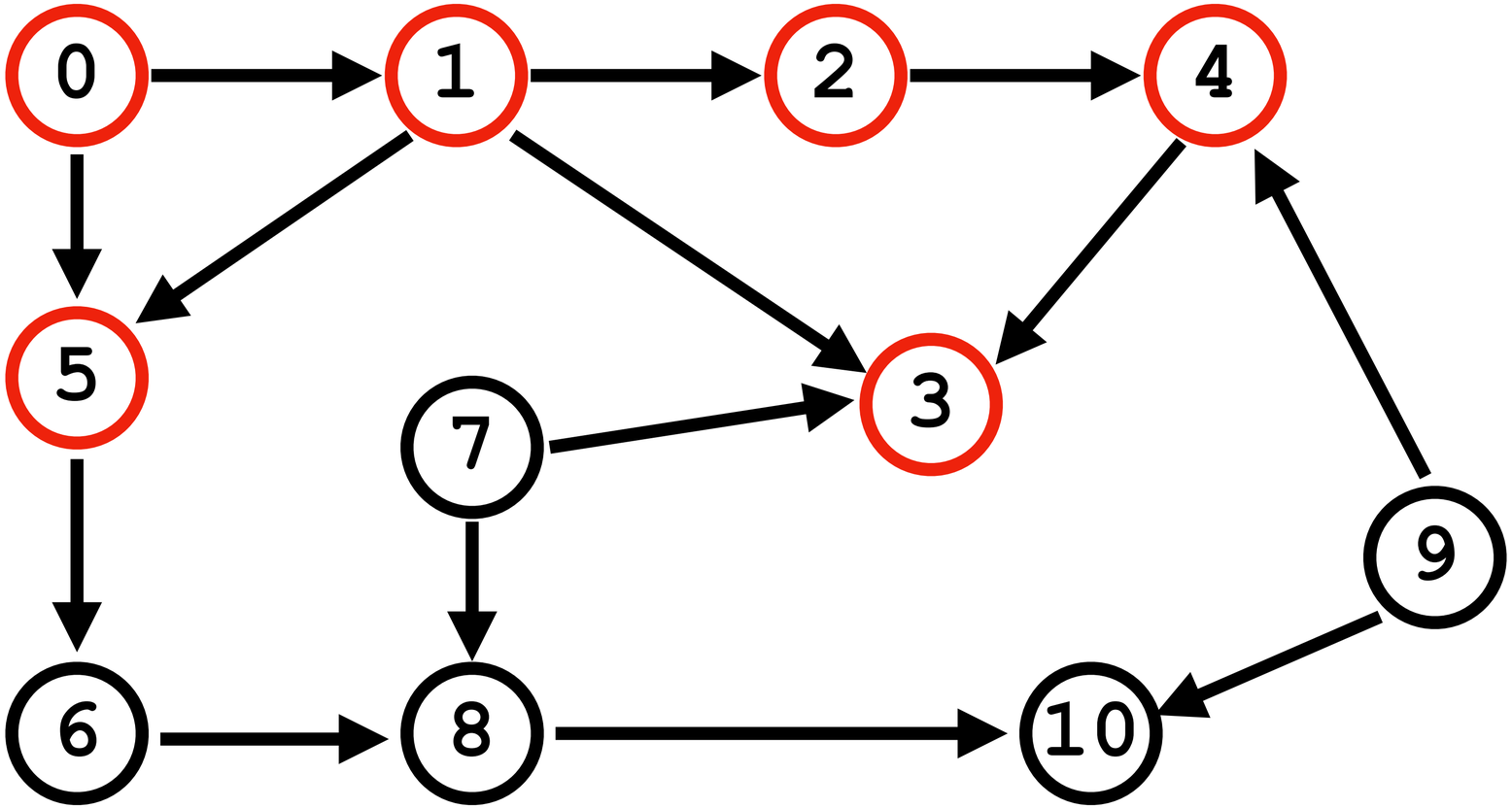
0 1 2 4

DFS Tracing



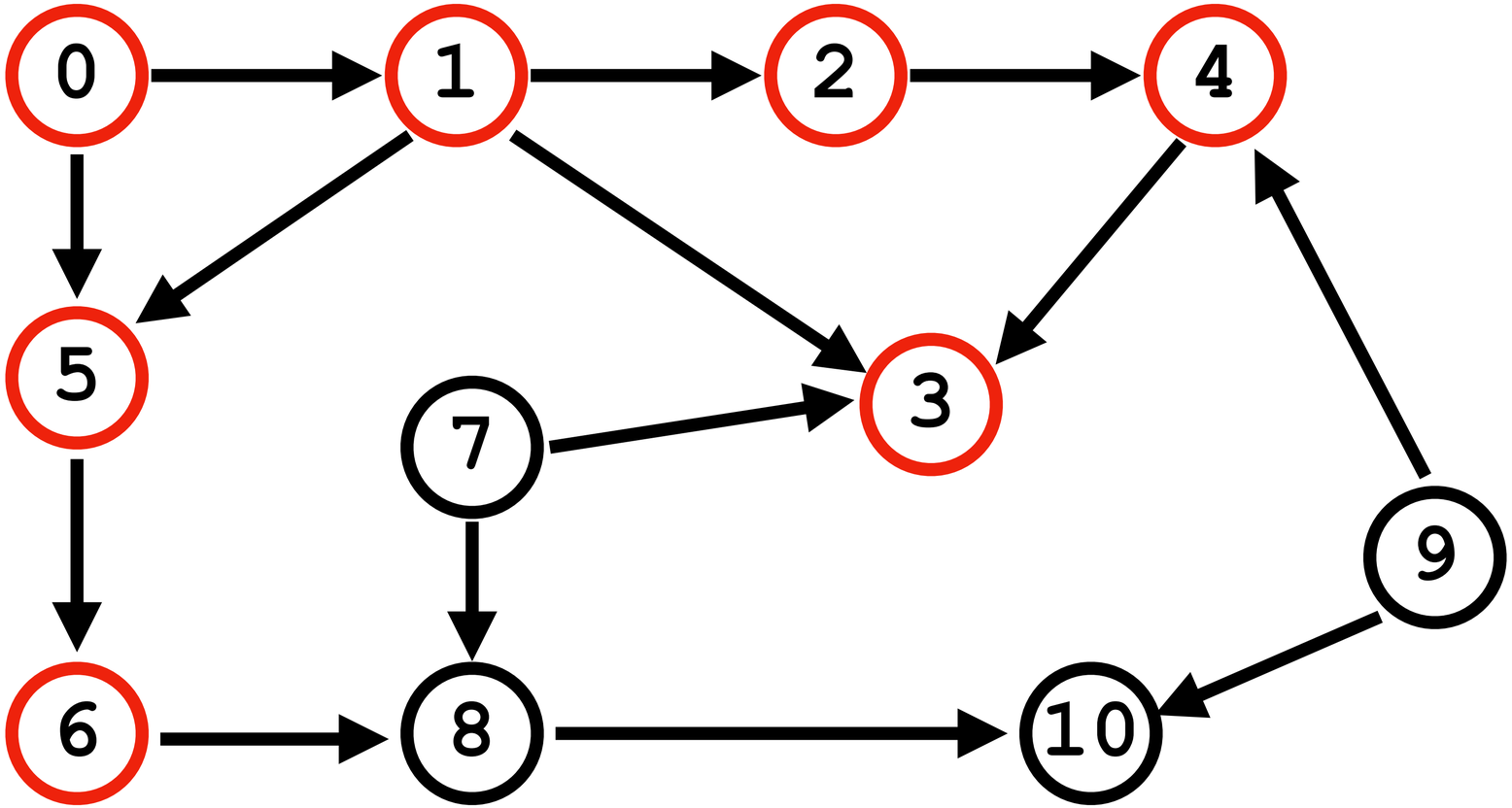
0 1 2 4 3

DFS Tracing



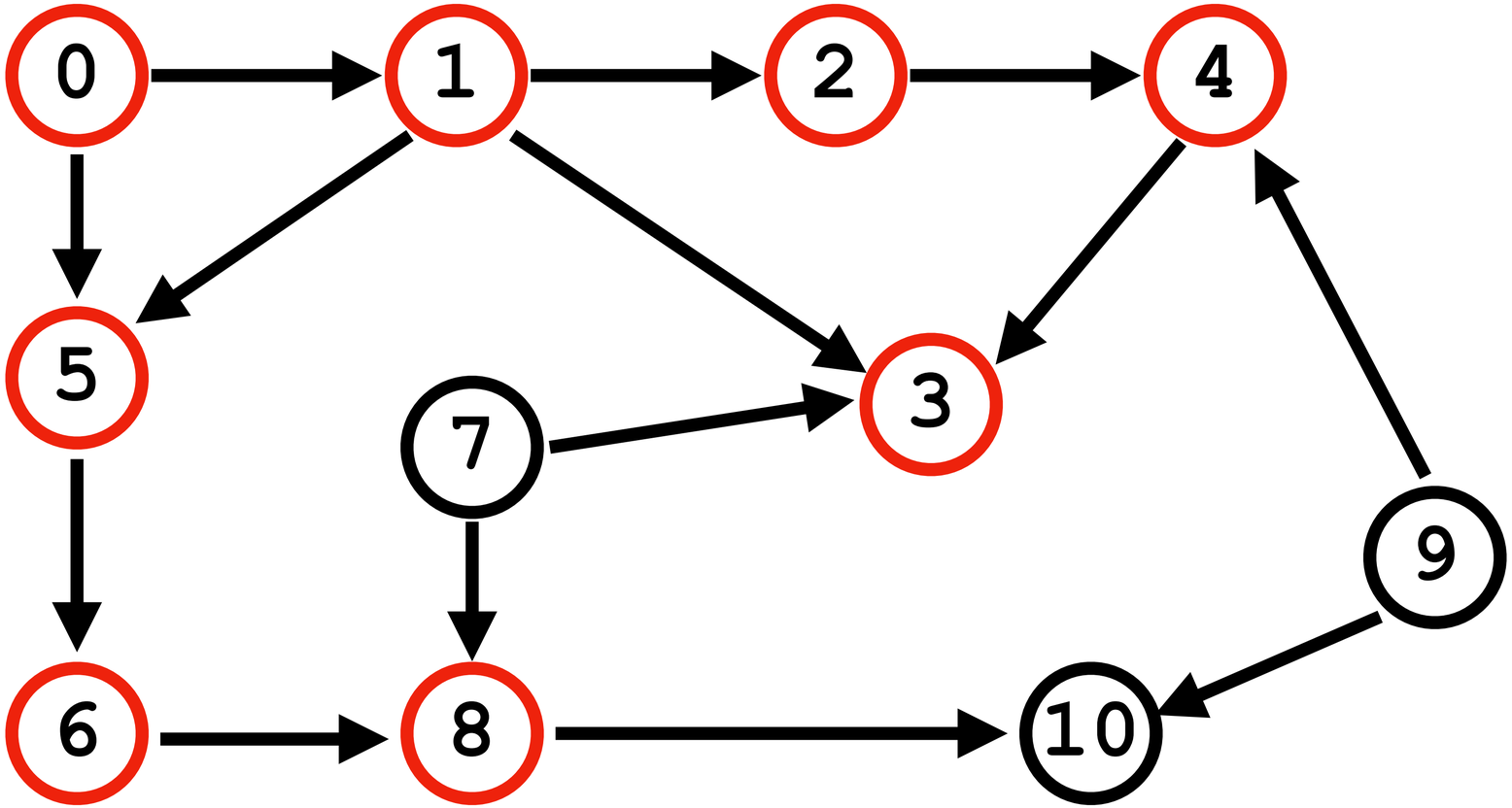
0 1 2 4 3 5

DFS Tracing



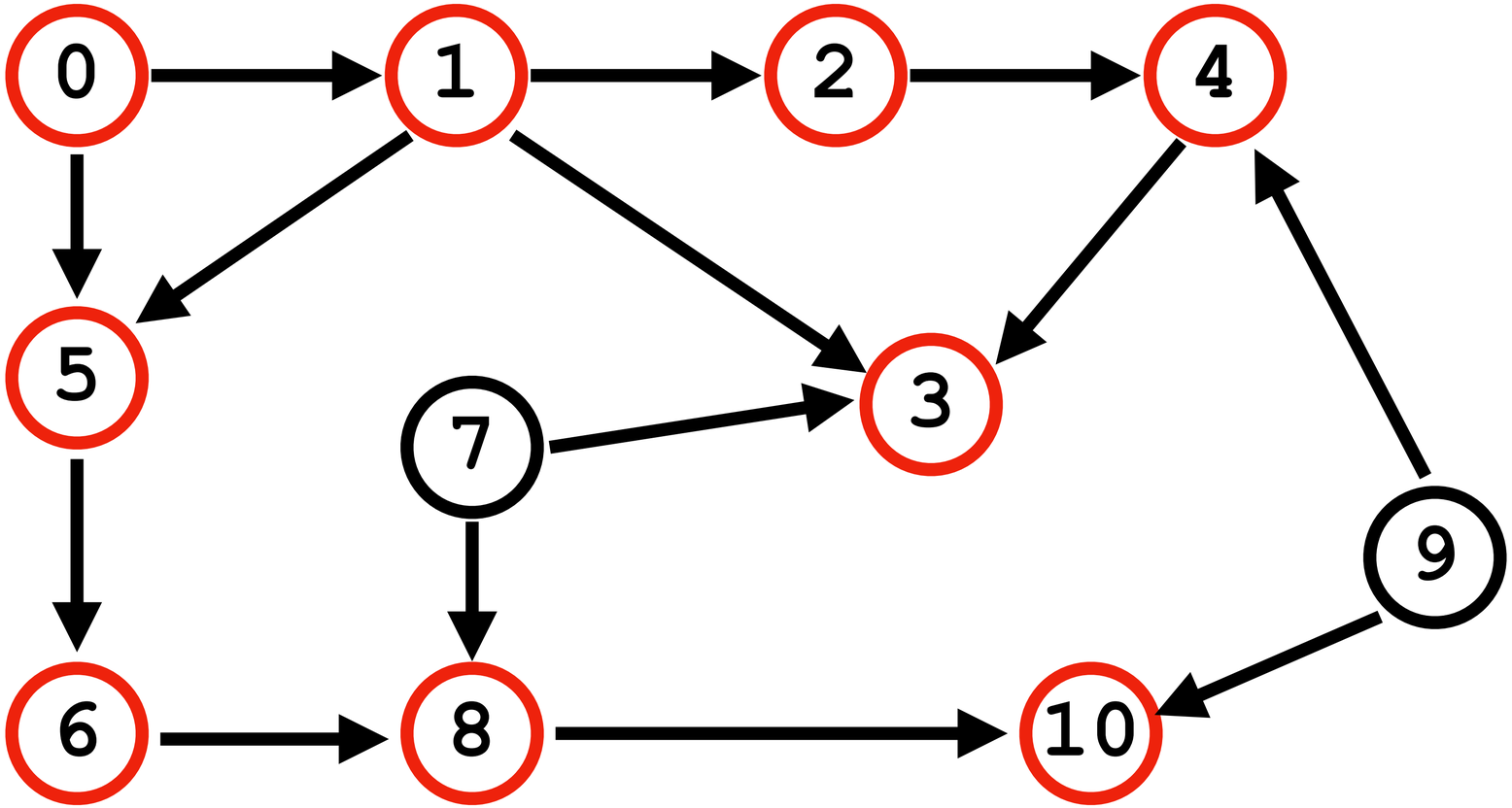
0 1 2 4 3 5 6

DFS Tracing



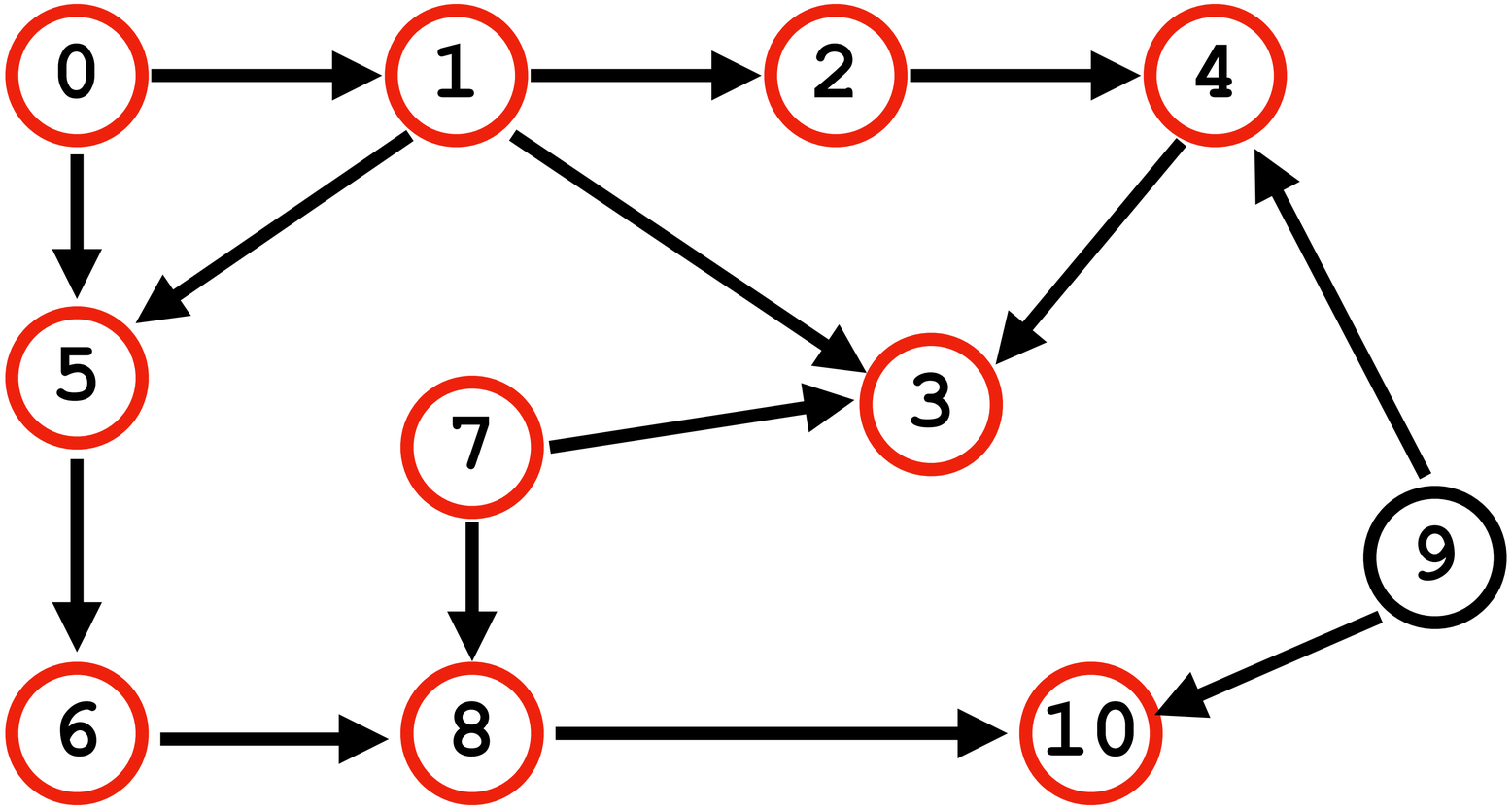
0 1 2 4 3 5 6 8

DFS Tracing



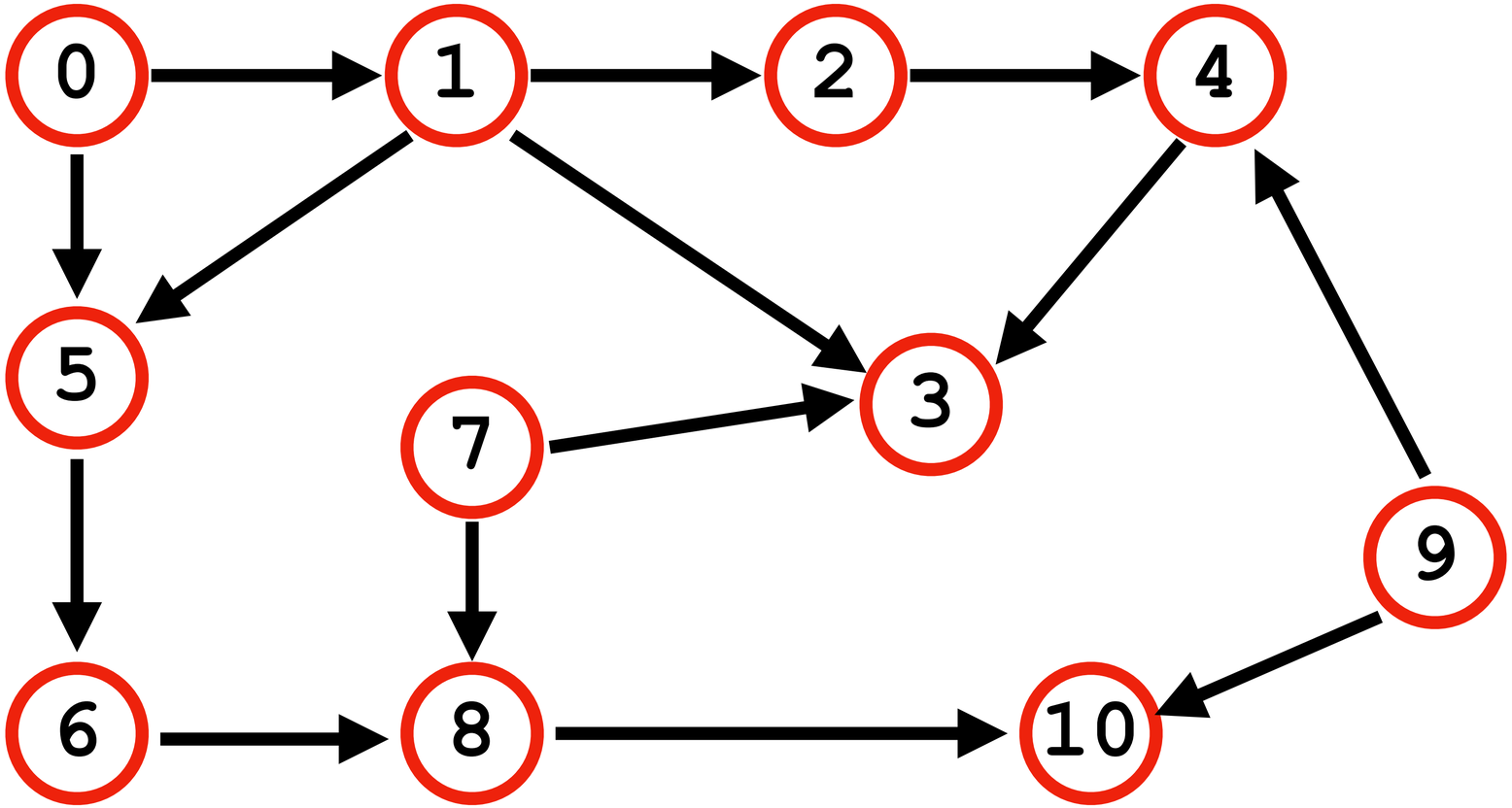
0 1 2 4 3 5 6 8 10

DFS Tracing



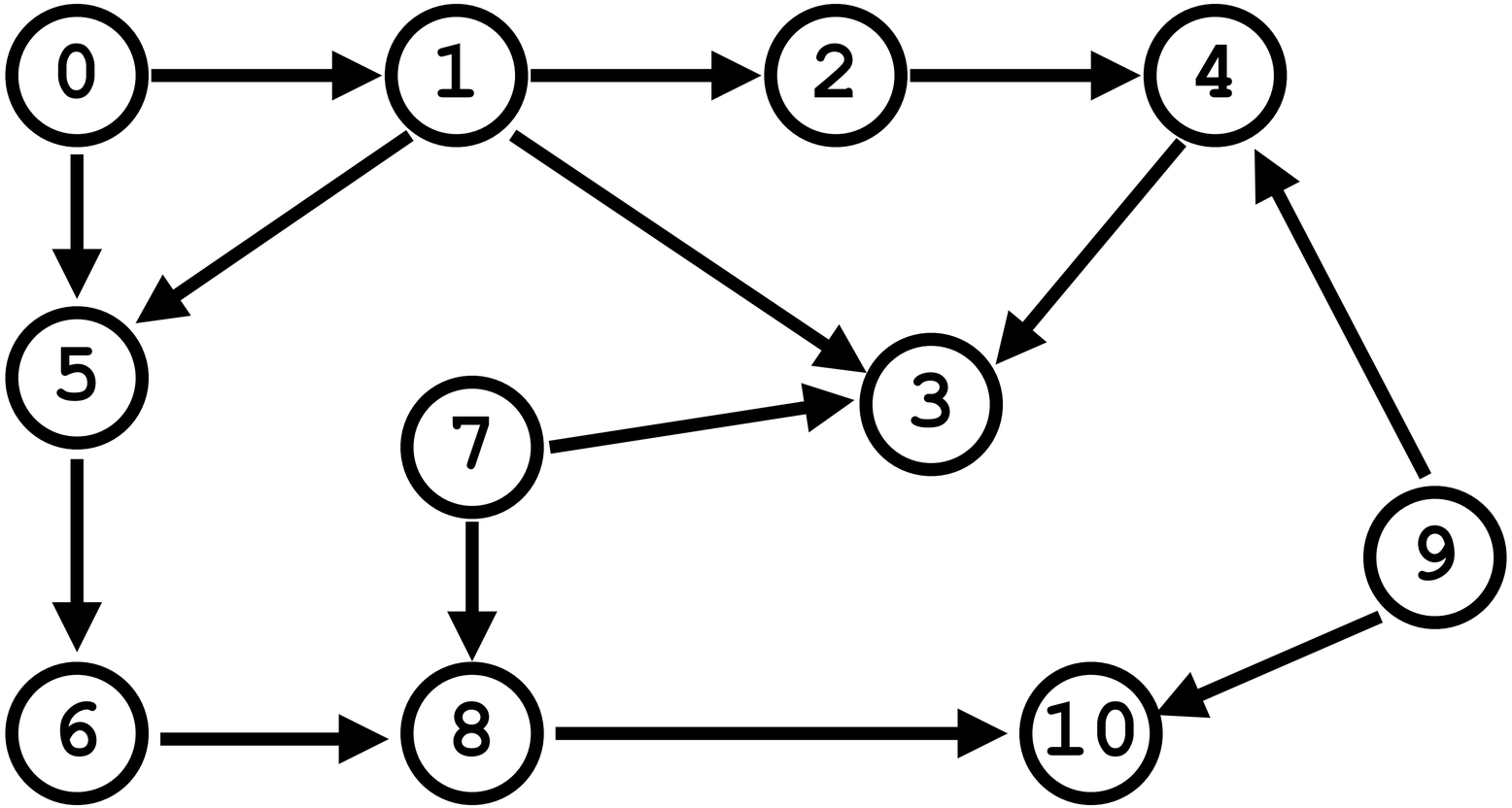
0 1 2 4 3 5 6 8 10 7

DFS Tracing



0 1 2 4 3 5 6 8 10 7 9

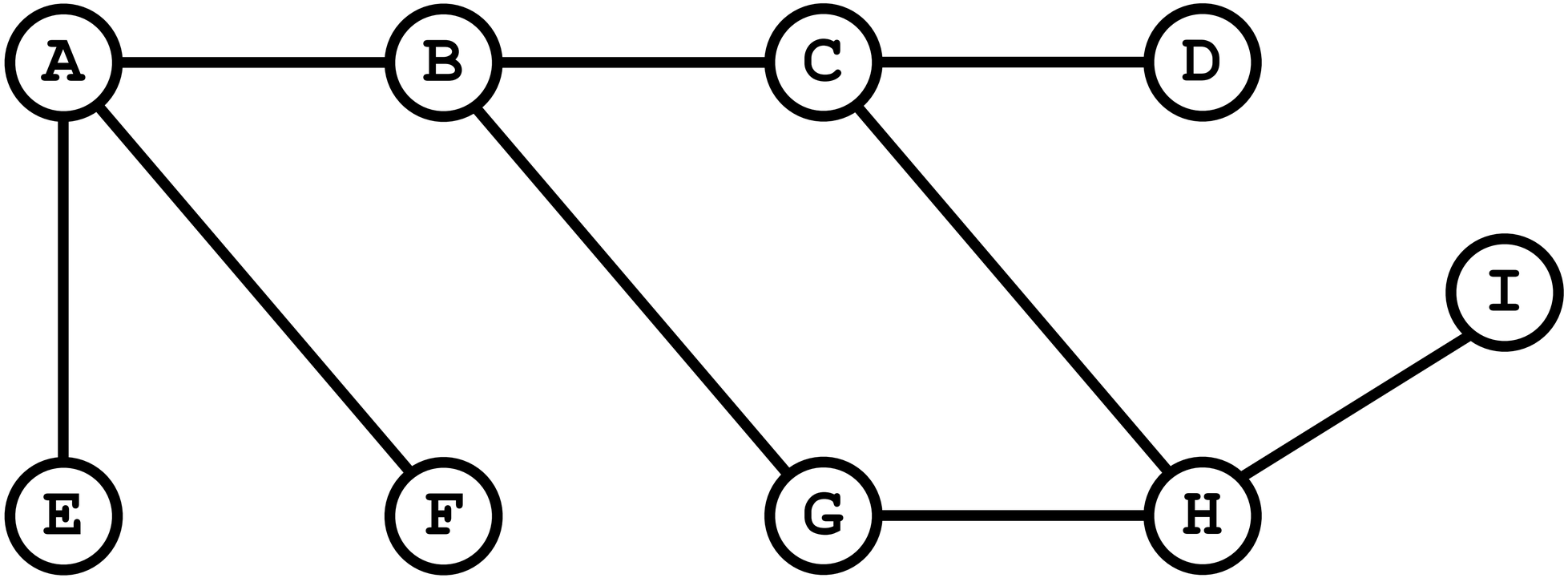
DFS Tracing - Completed Traversal



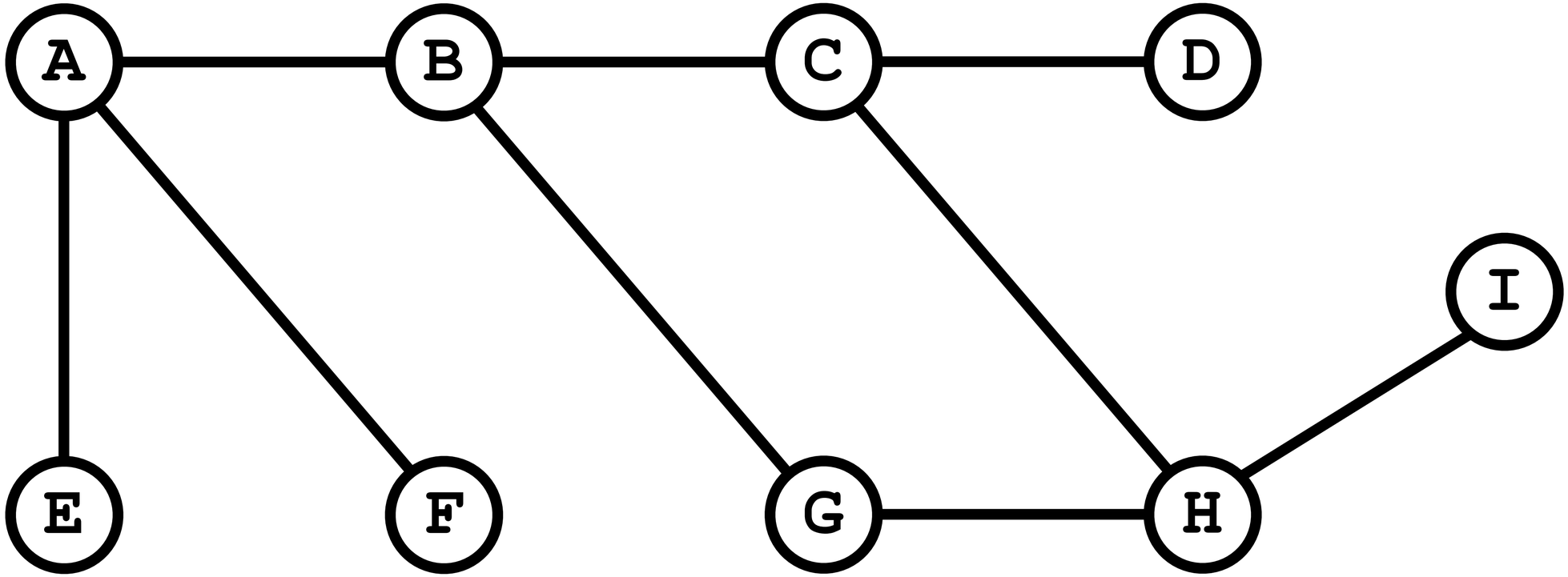
0 1 2 4 3 5 6 8 10 7 9

DFS Tracing - Practice

Perform DFS tracing on the graph below starting from vertex `A`.

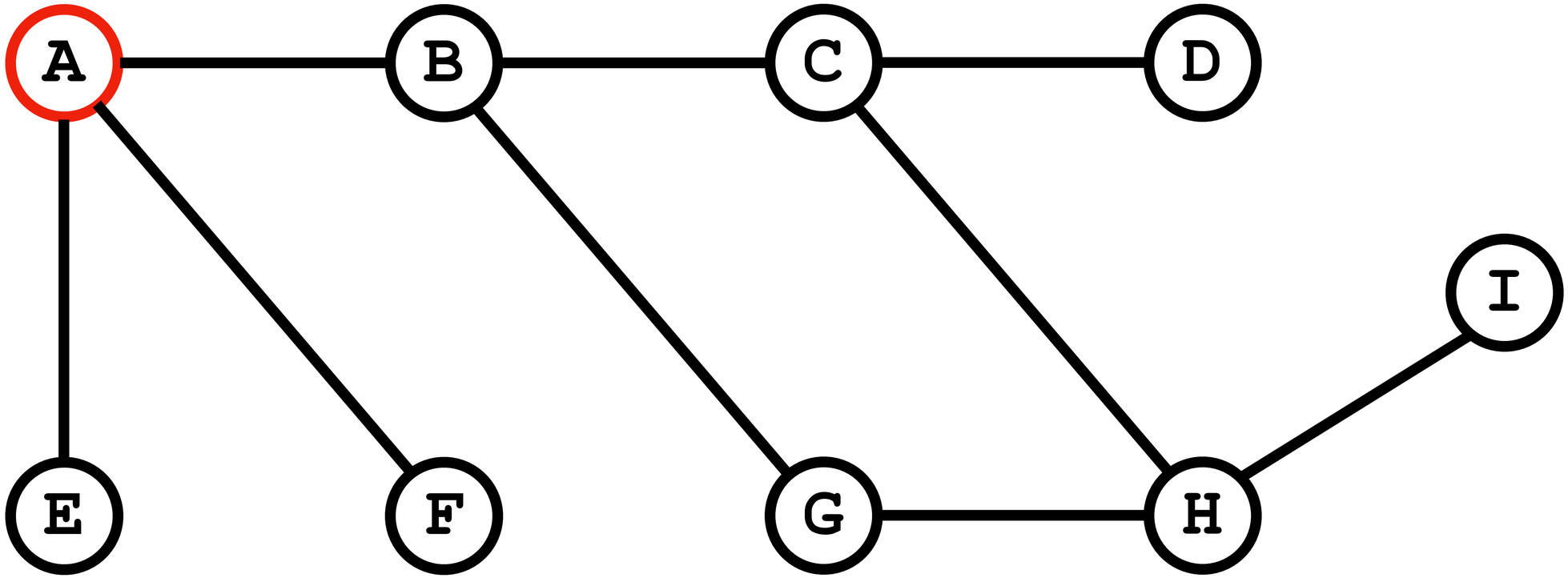


DFS Tracing - Practice (Answer)



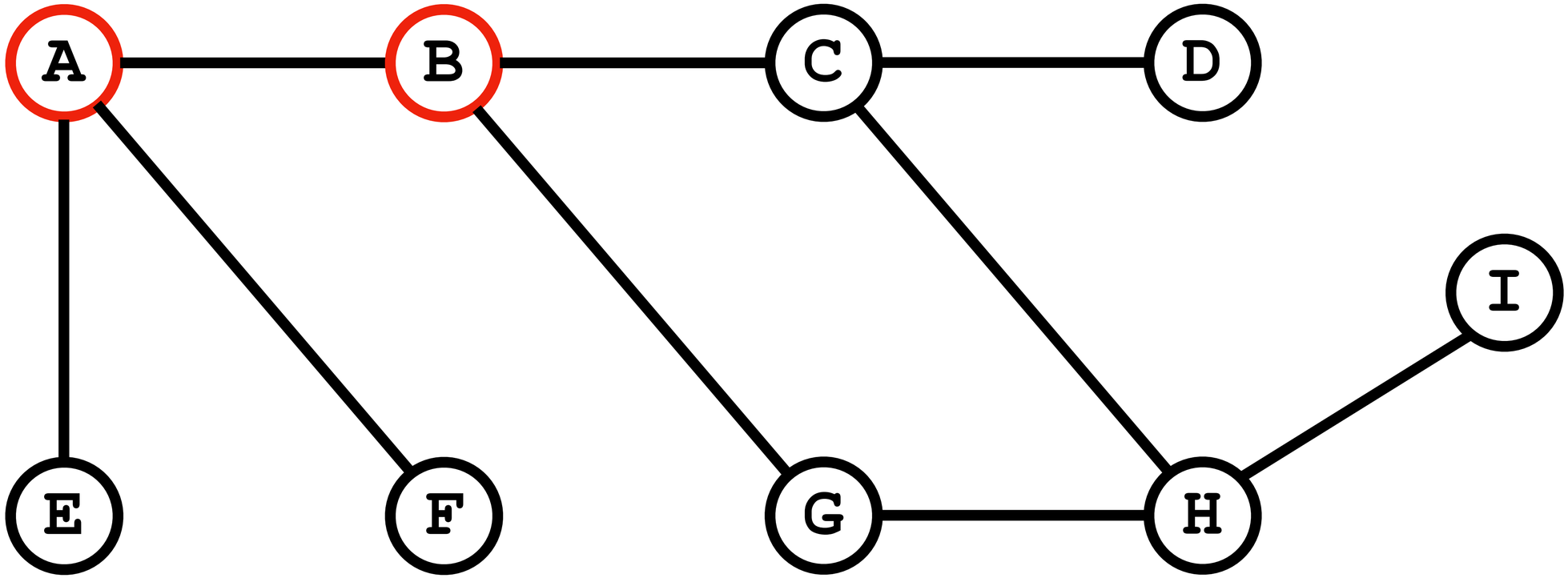
A B C D H G I E F

DFS Tracing - Practice



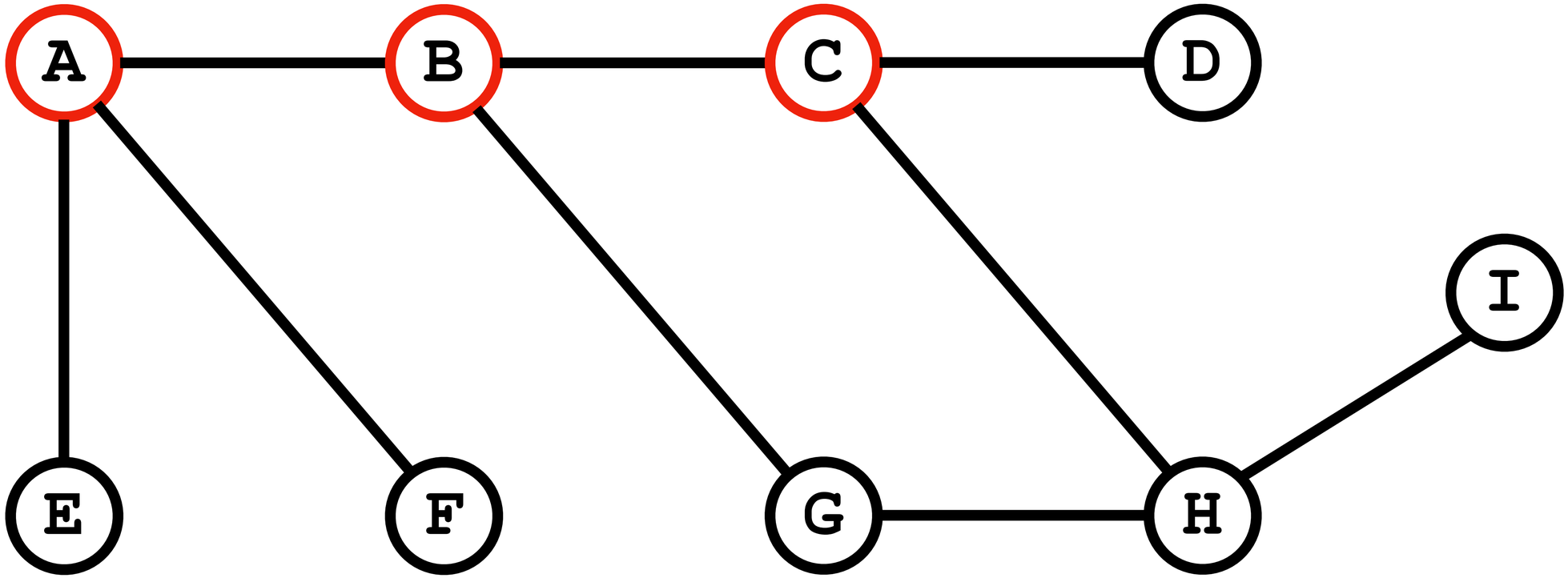
A

DFS Tracing - Practice



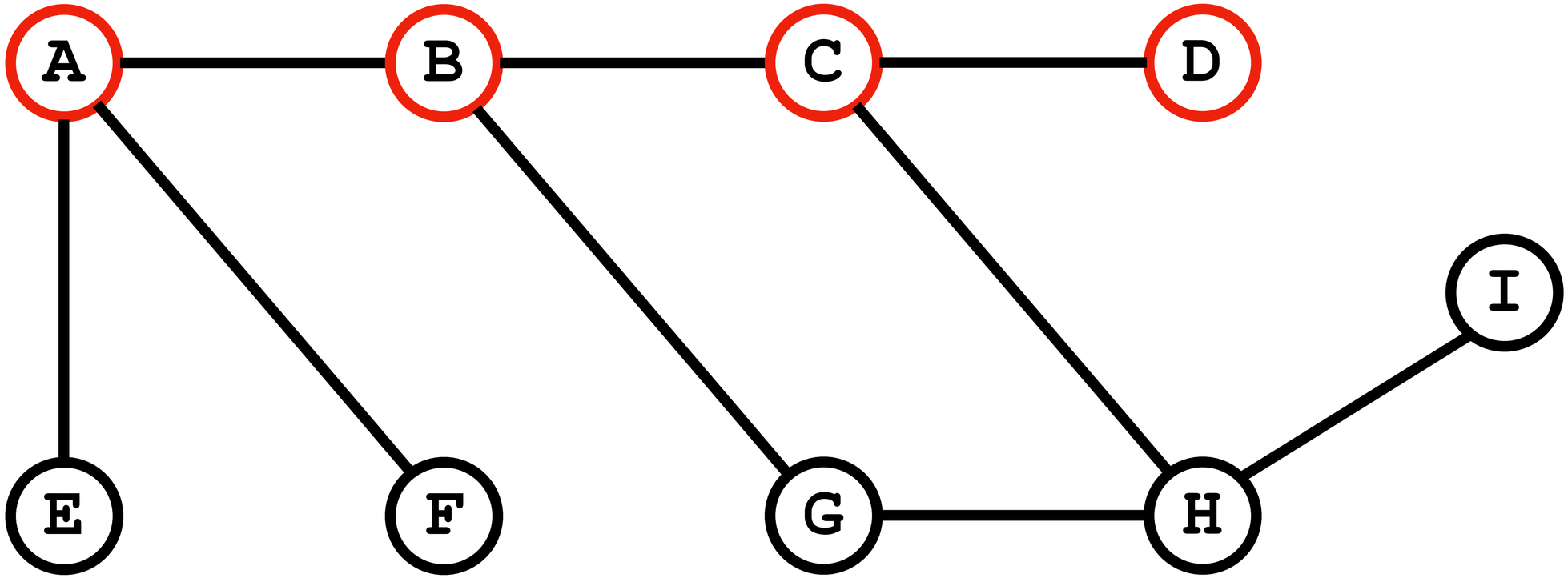
A B

DFS Tracing - Practice



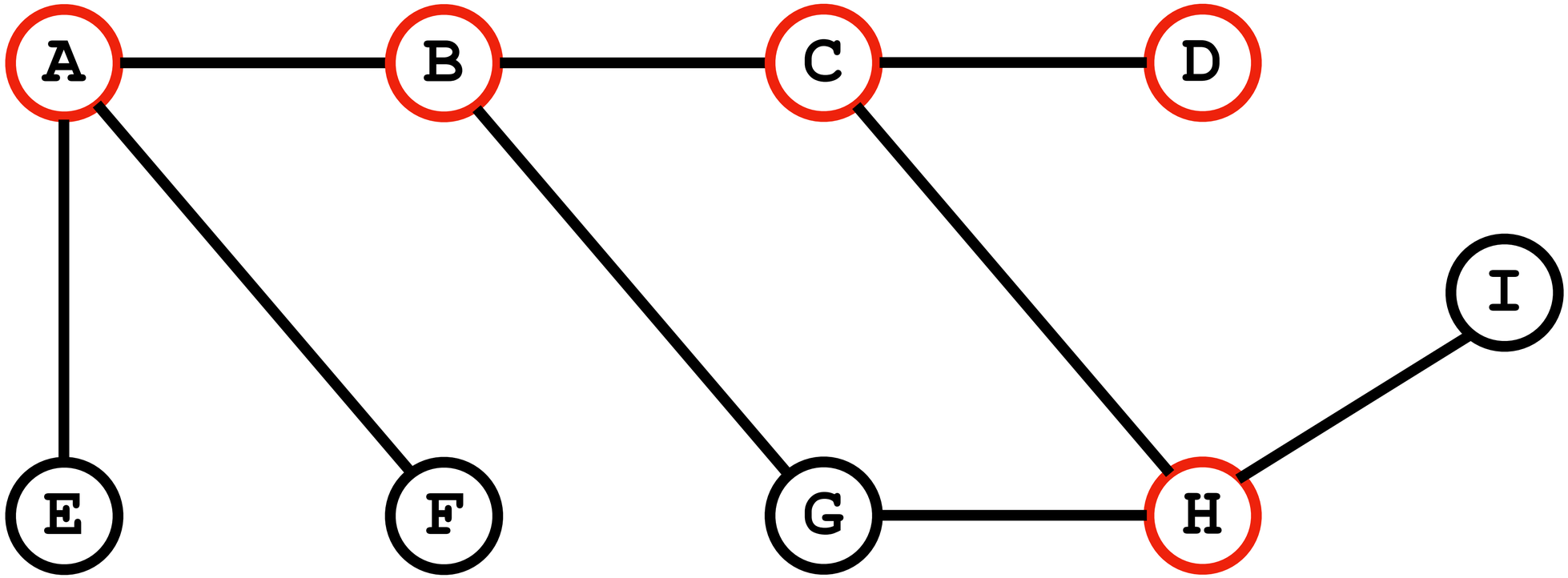
A B C

DFS Tracing - Practice



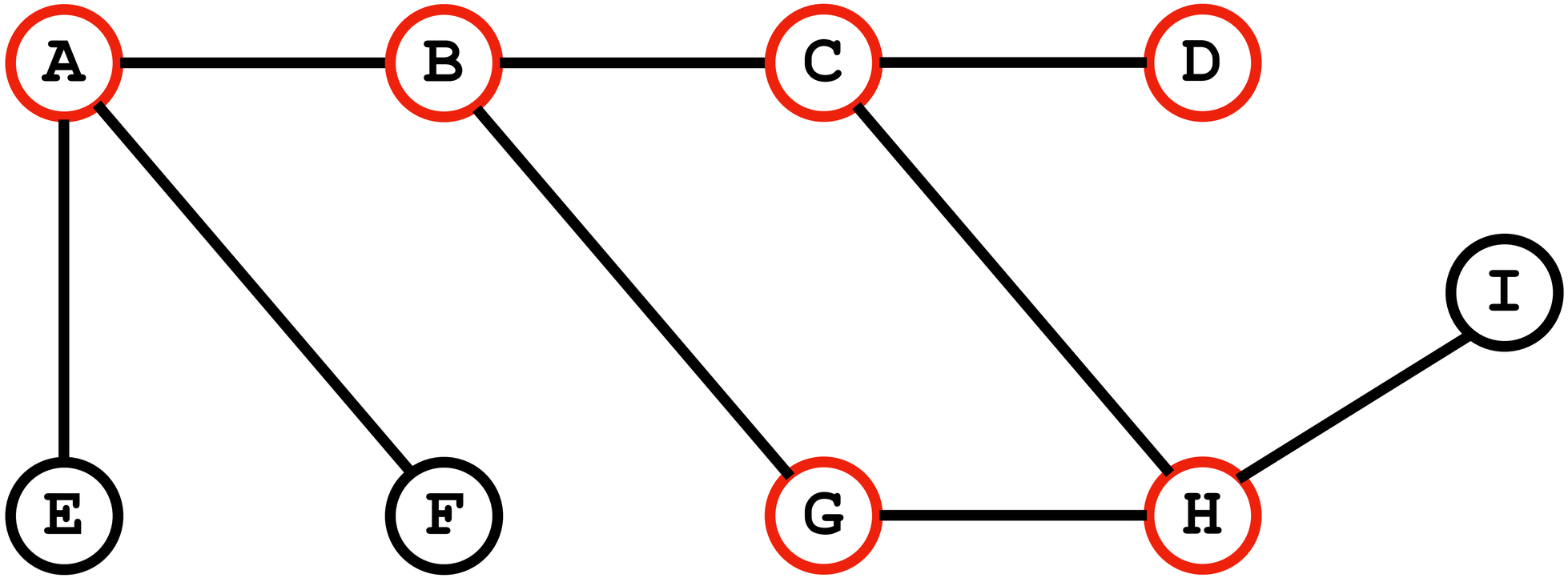
A B C D

DFS Tracing - Practice



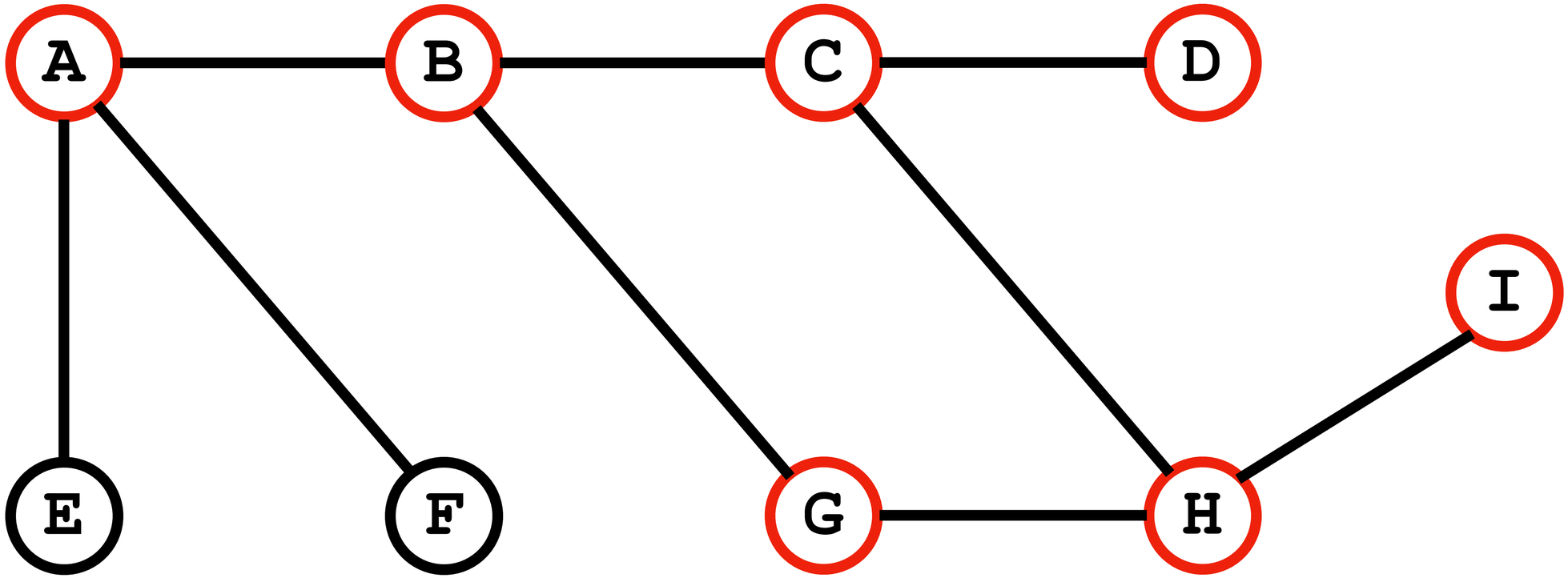
A B C D H

DFS Tracing - Practice



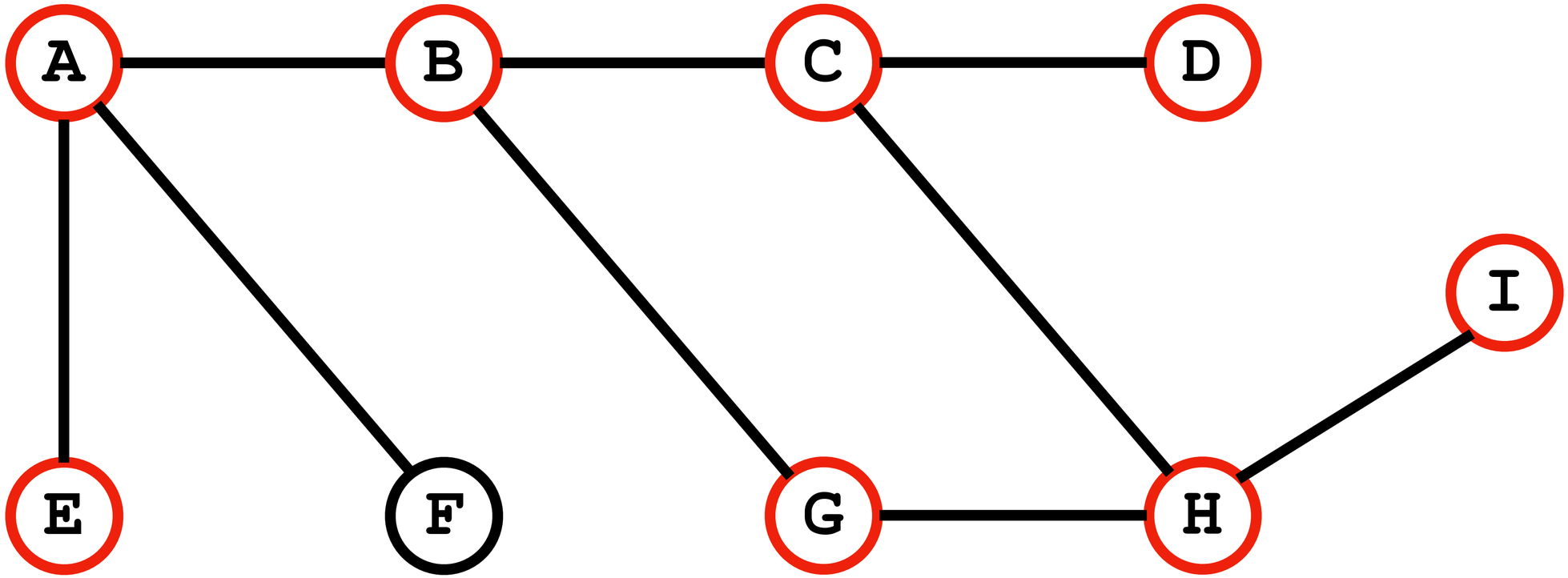
A B C D H G

DFS Tracing - Practice



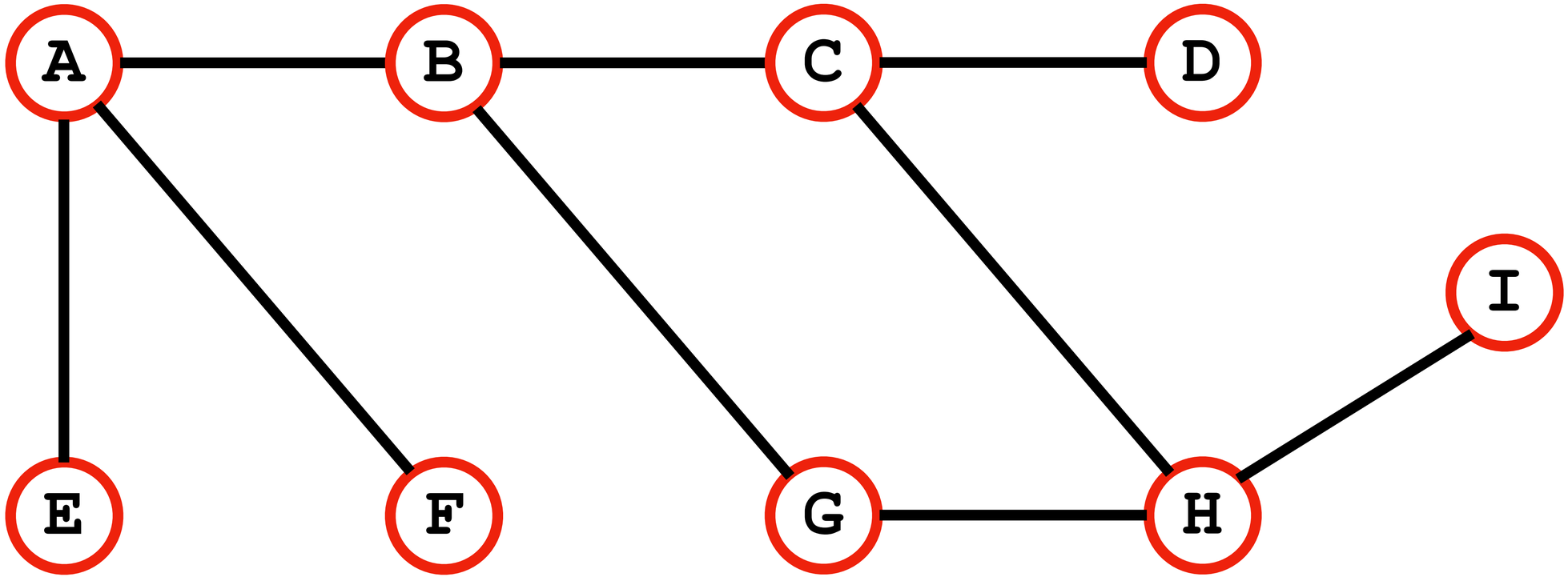
A B C D H G I

DFS Tracing - Practice



A B C D H G I E

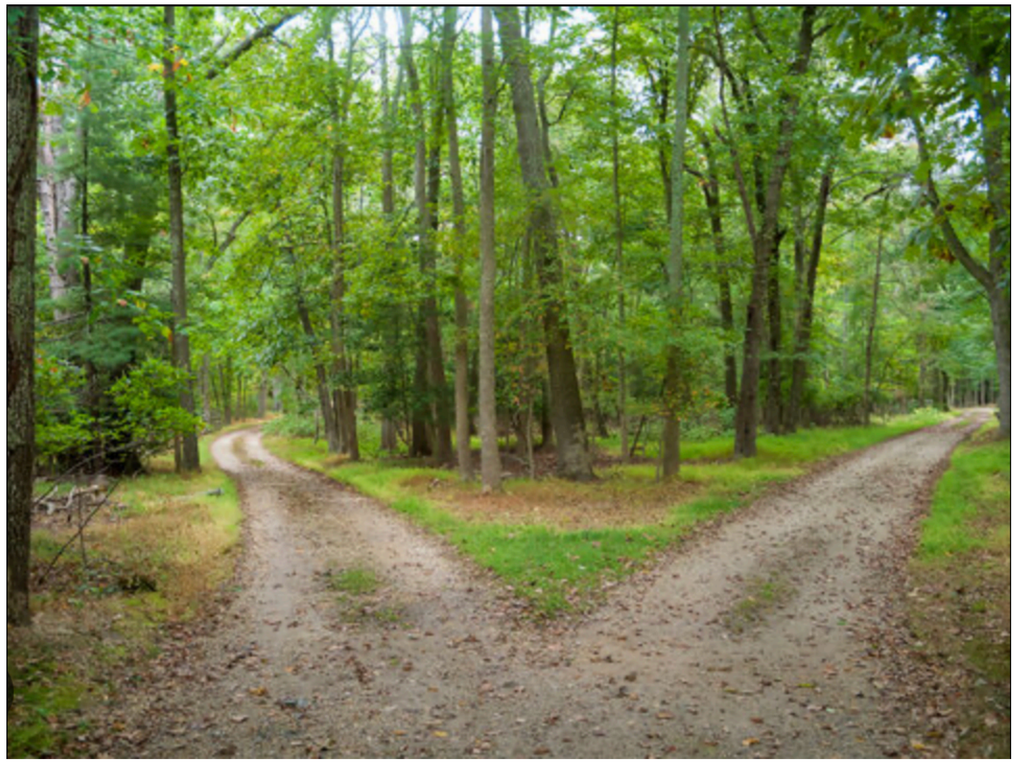
DFS Tracing - Practice



A B C D H G I E F

Breadth First Search (BFS)

- BFS is a graph traversal algorithm
- BFS explores a graph by “level”, visiting all nodes at the current level before moving to the next level
- BFS is implemented using a queue (to maintain order of node exploration)

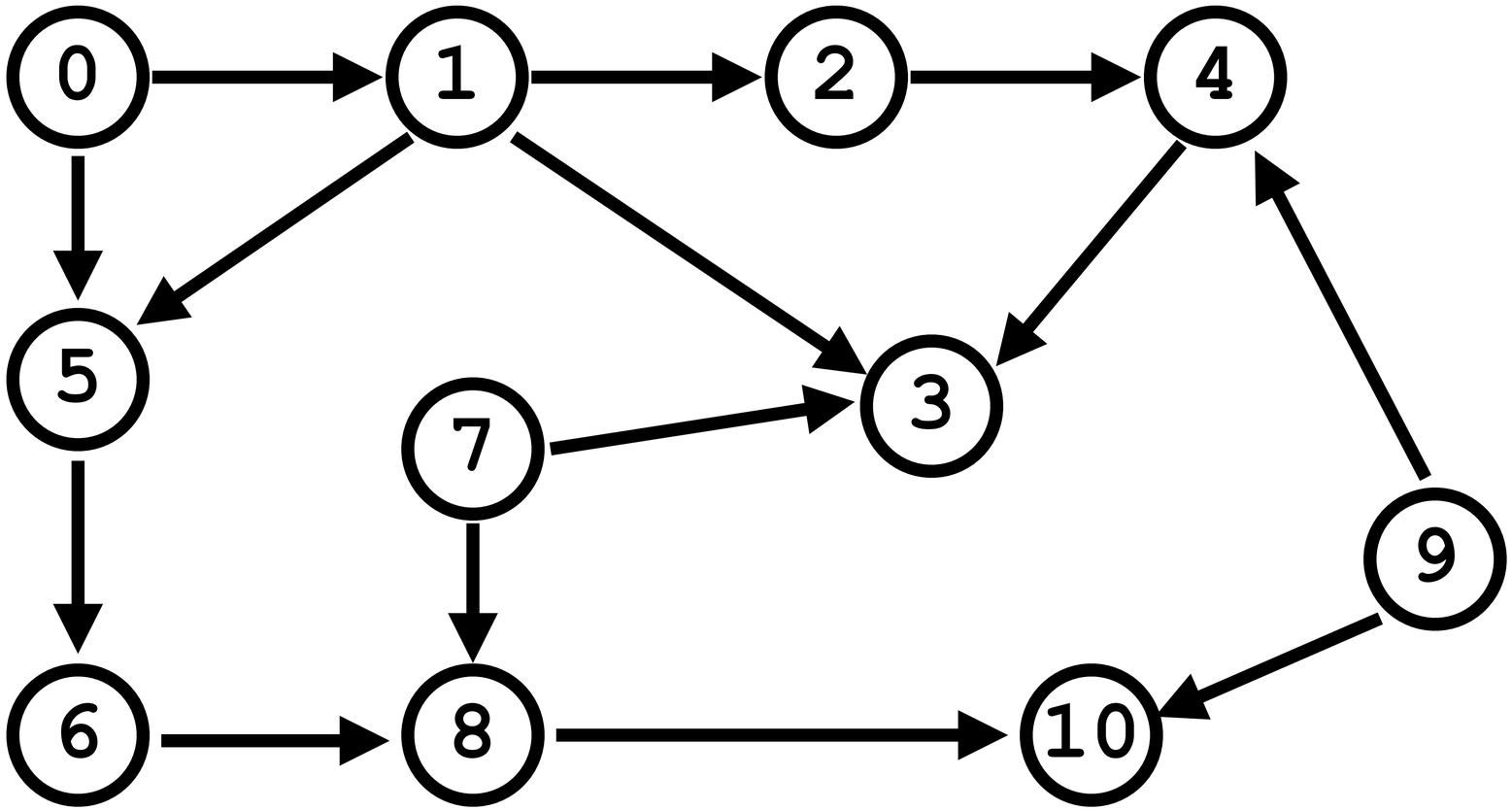


BFS Algorithm

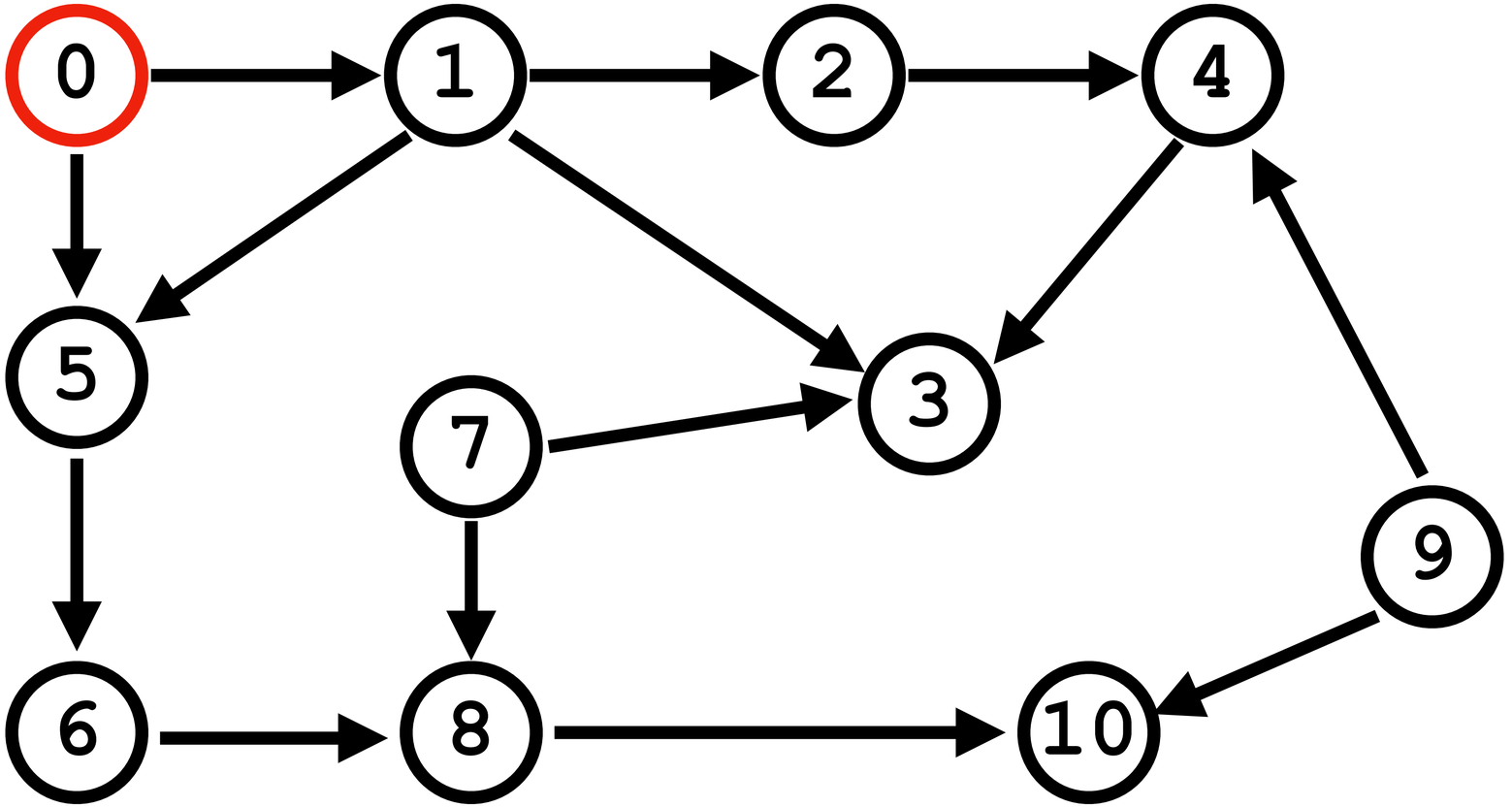
1. for each vertex v in the graph
       if v is not visited
          add v to the queue
2. mark v as visited
3. while the queue is not empty
       remove vertex u from the queue
       retrieve the vertices adjacent to u
       for each vertex w that is adjacent to u
           add w to the queue
           mark w as visited

BFS Tracing

Let’s trace the breadth first search of the graph below staring from veretx 0

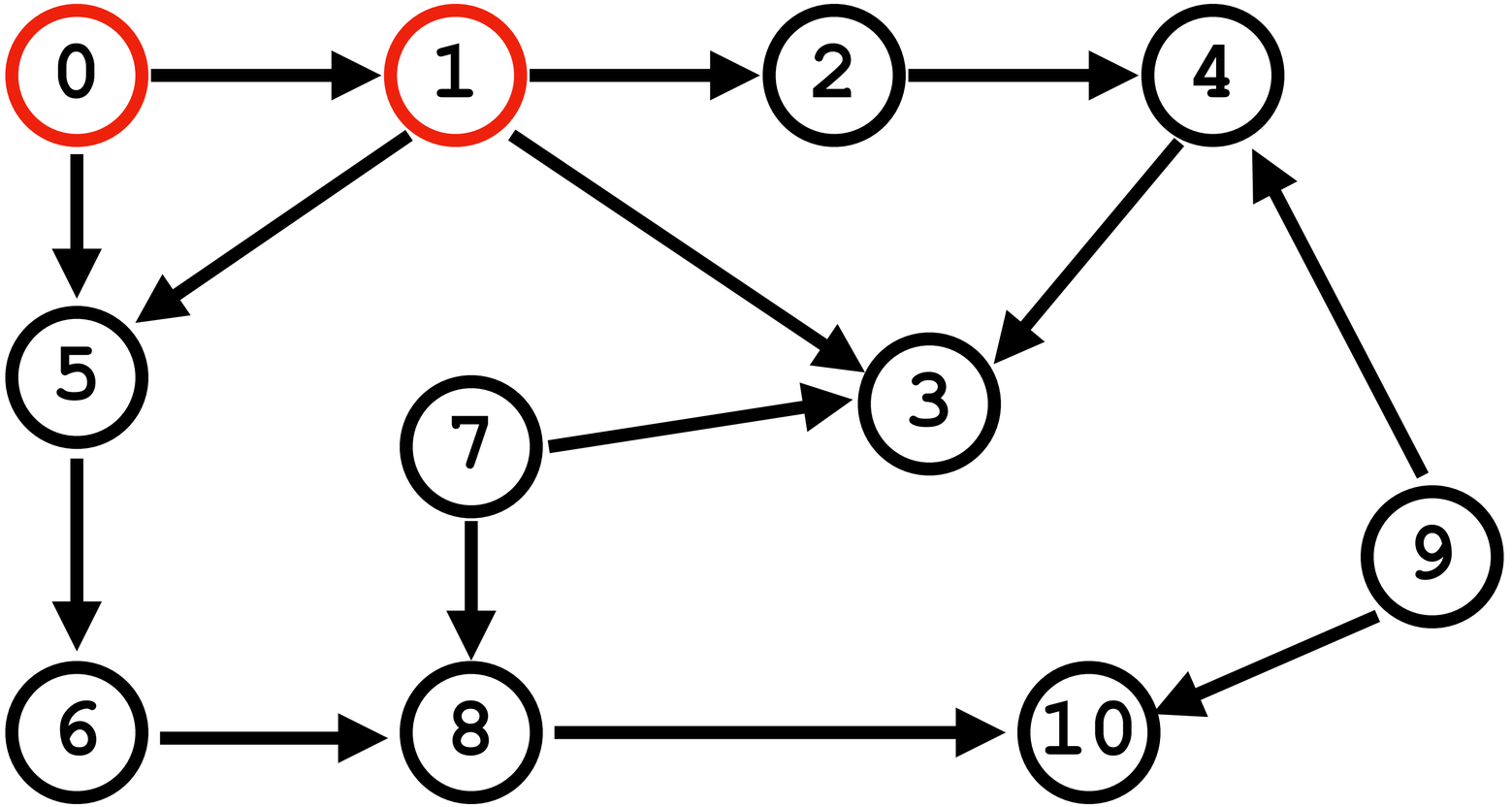


BFS Tracing



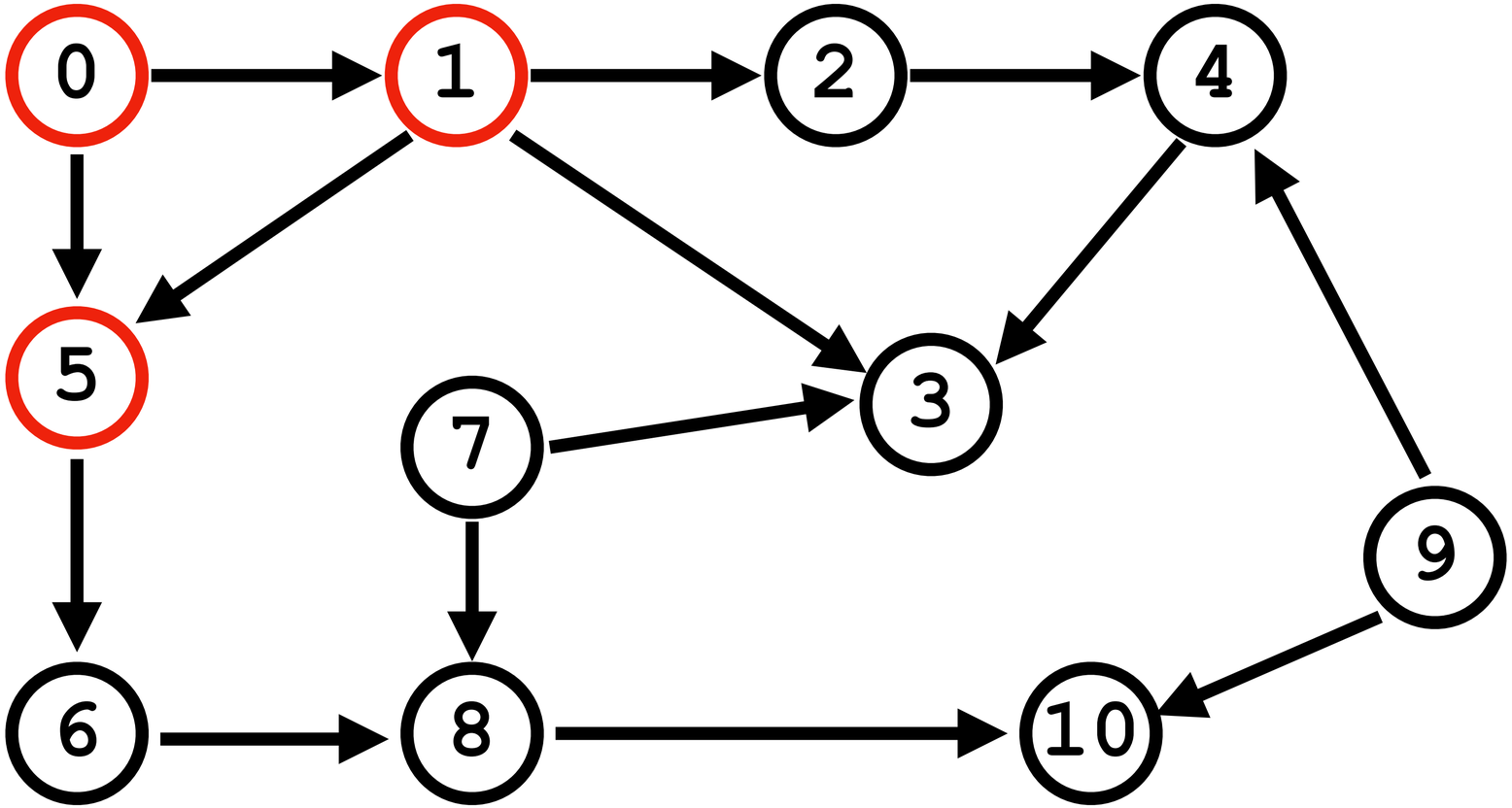
0

BFS Tracing



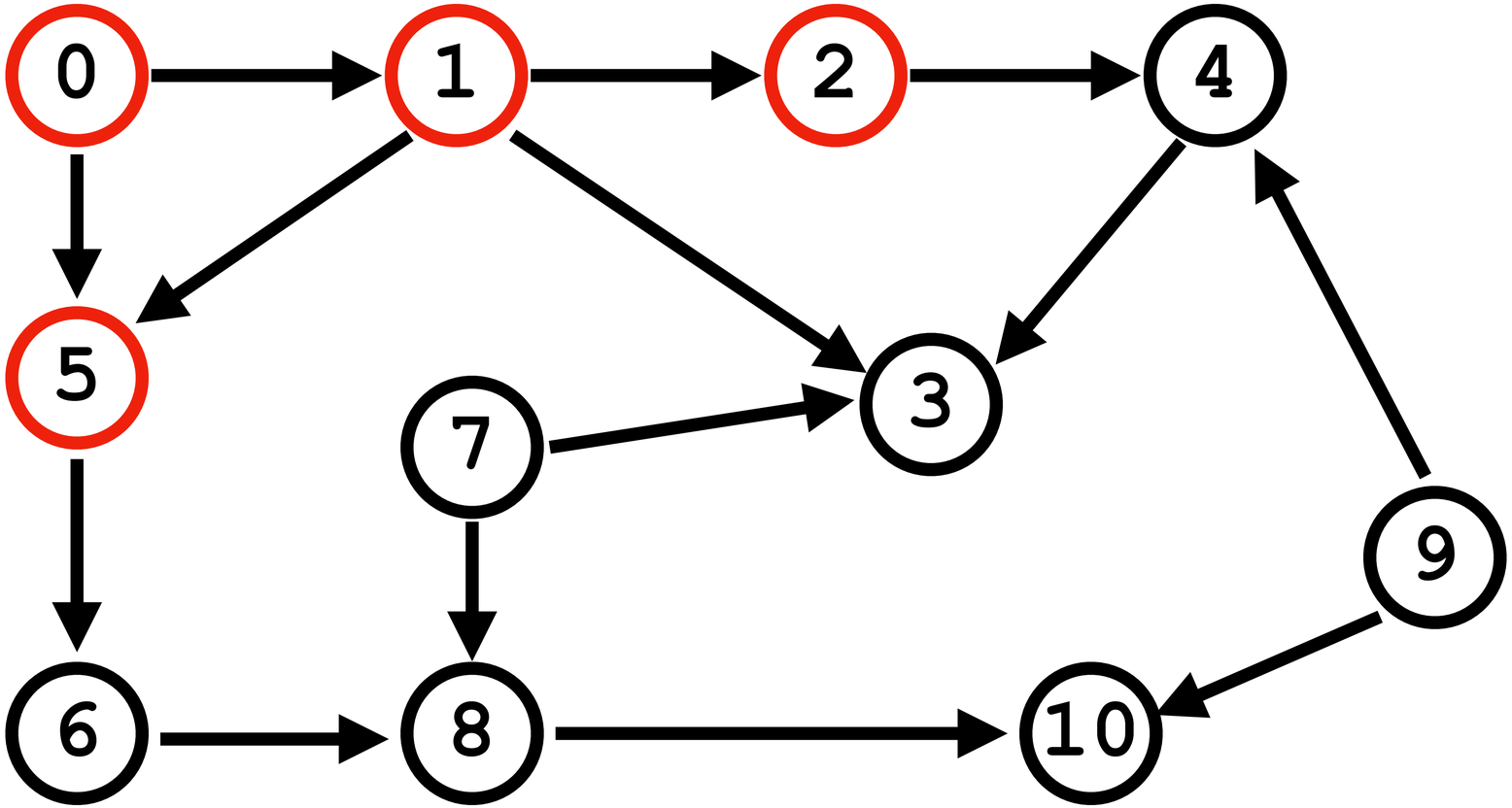
0 1

BFS Tracing



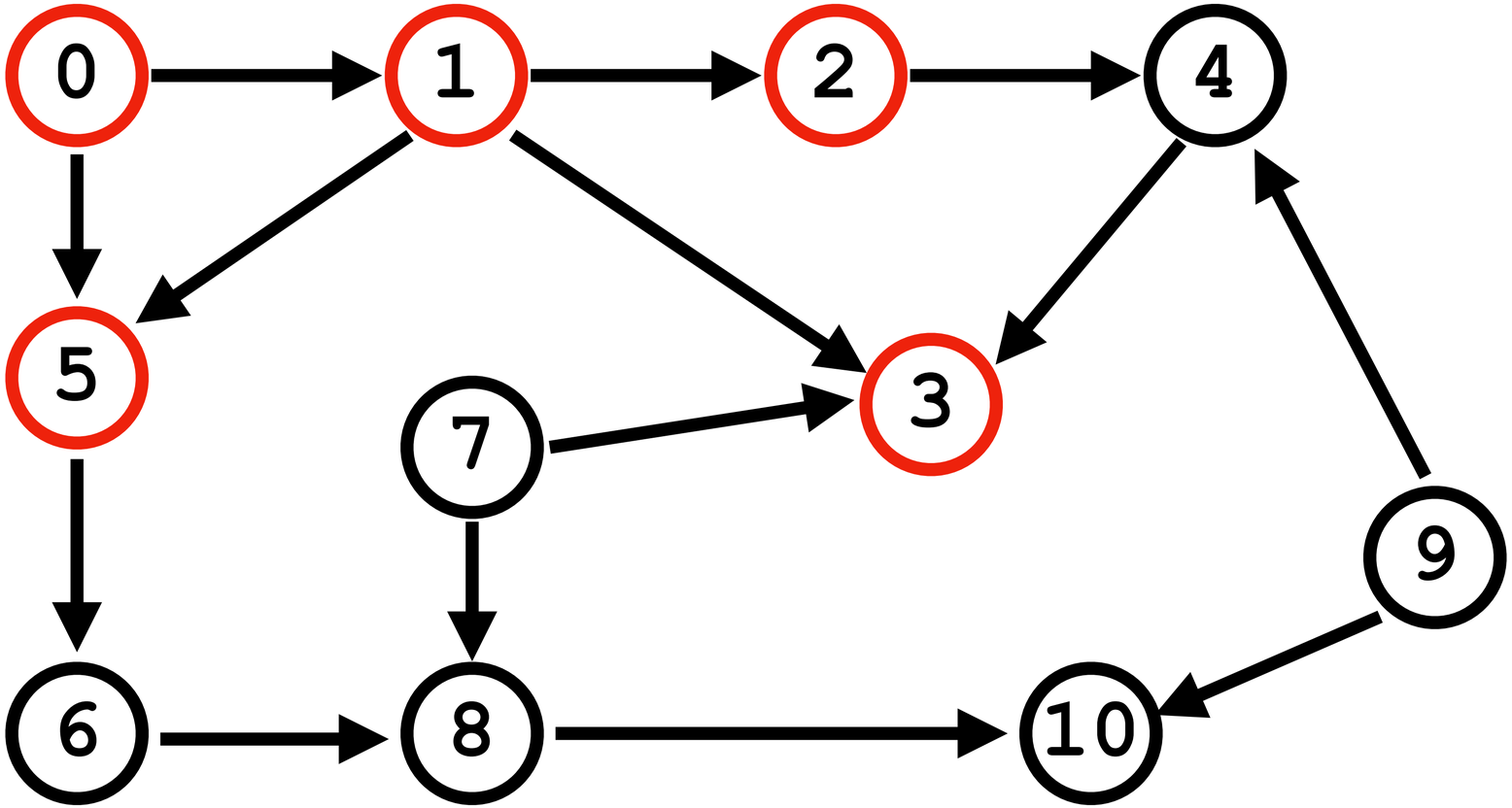
0 1 5

BFS Tracing



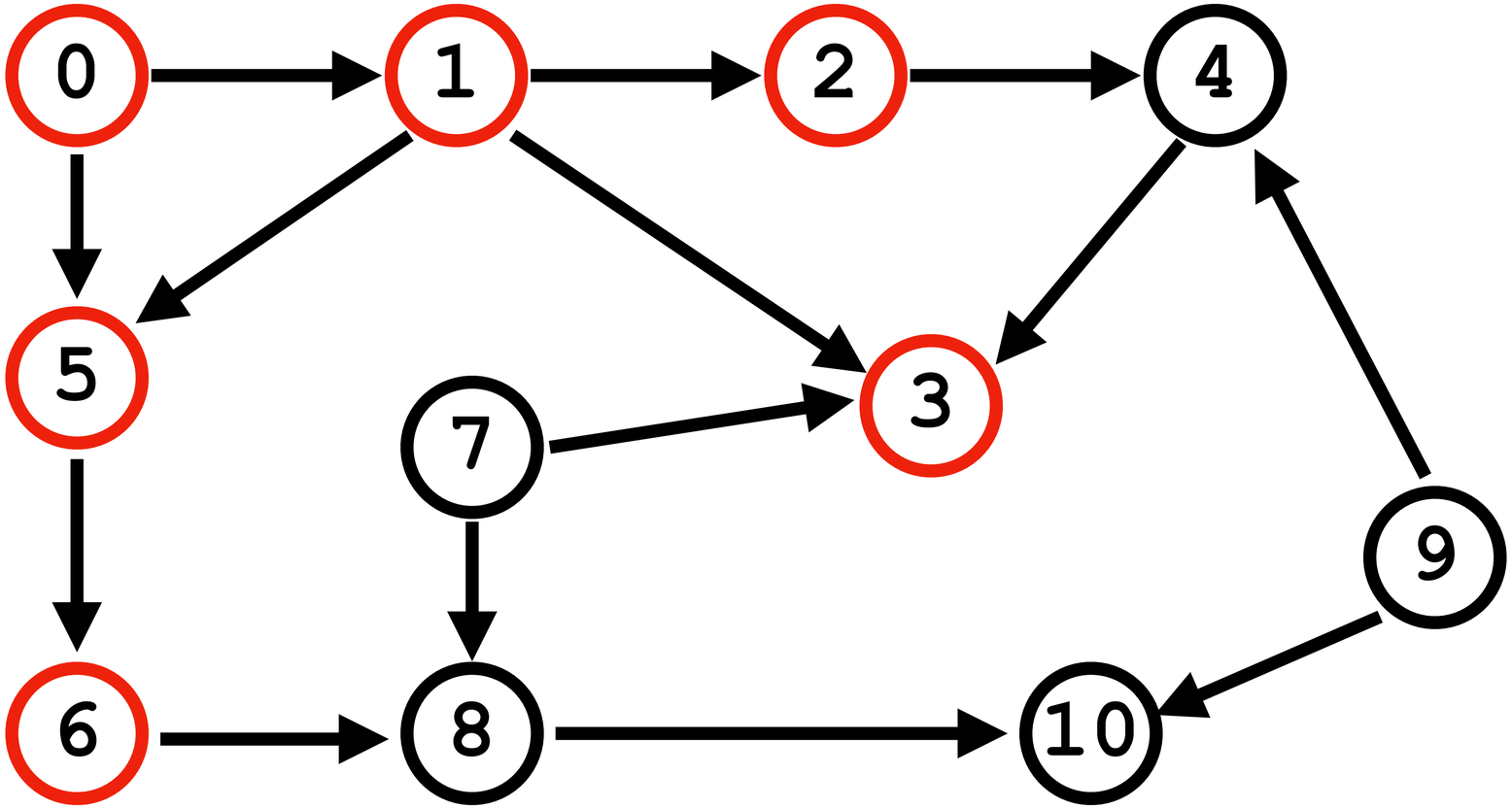
0 1 5 2

BFS Tracing



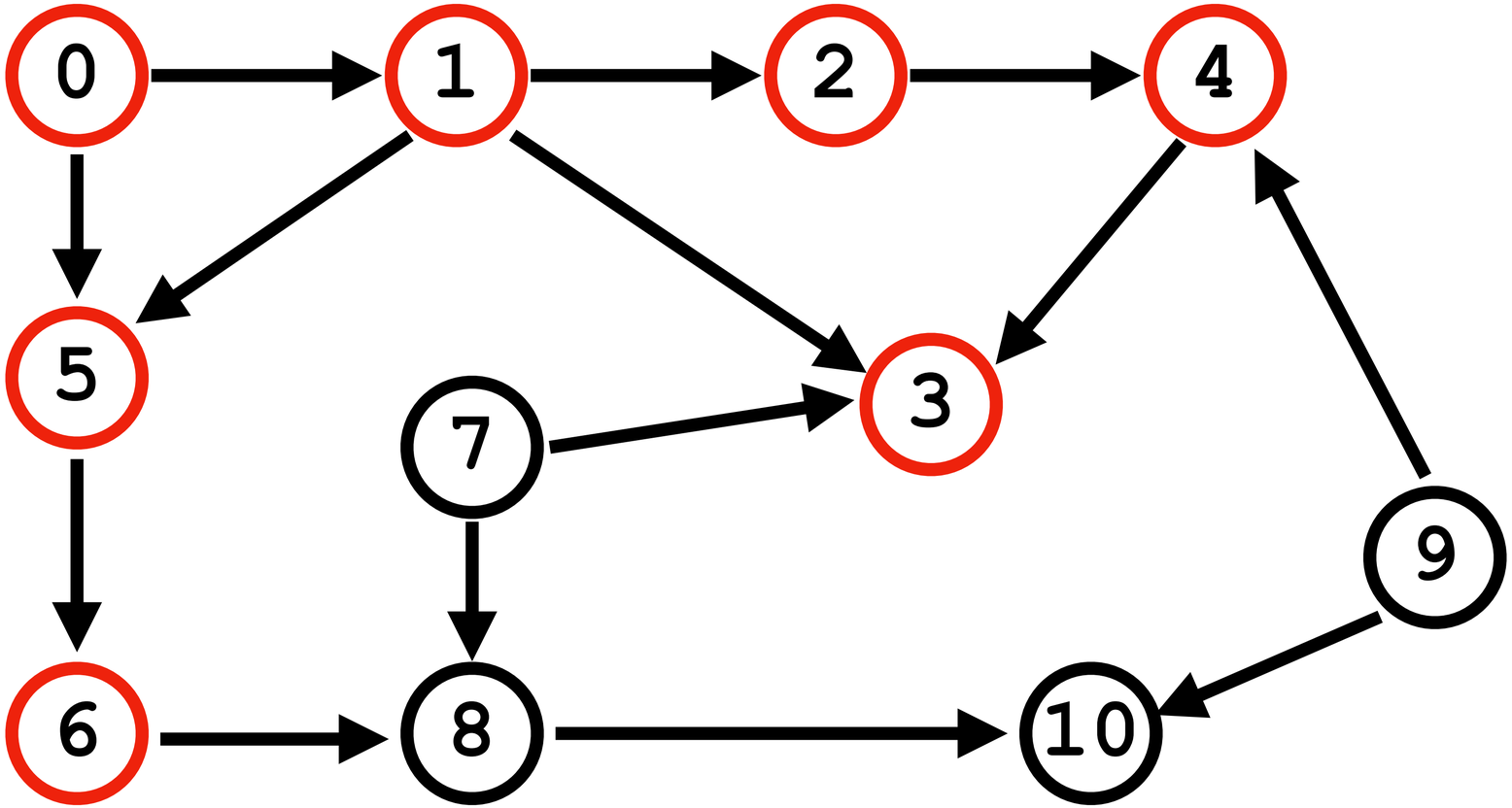
0 1 5 2 3

BFS Tracing



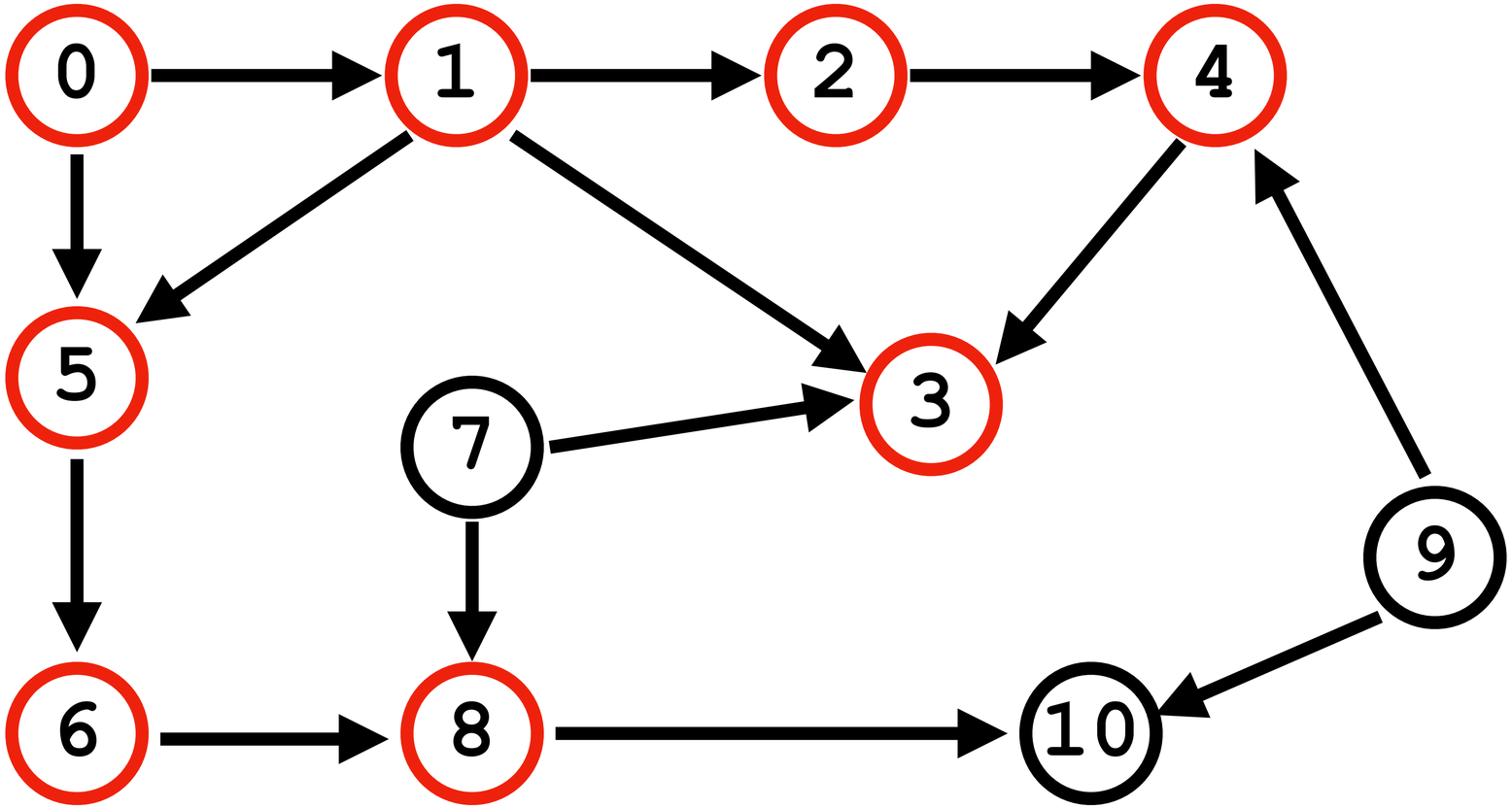
0 1 5 2 3 6

BFS Tracing



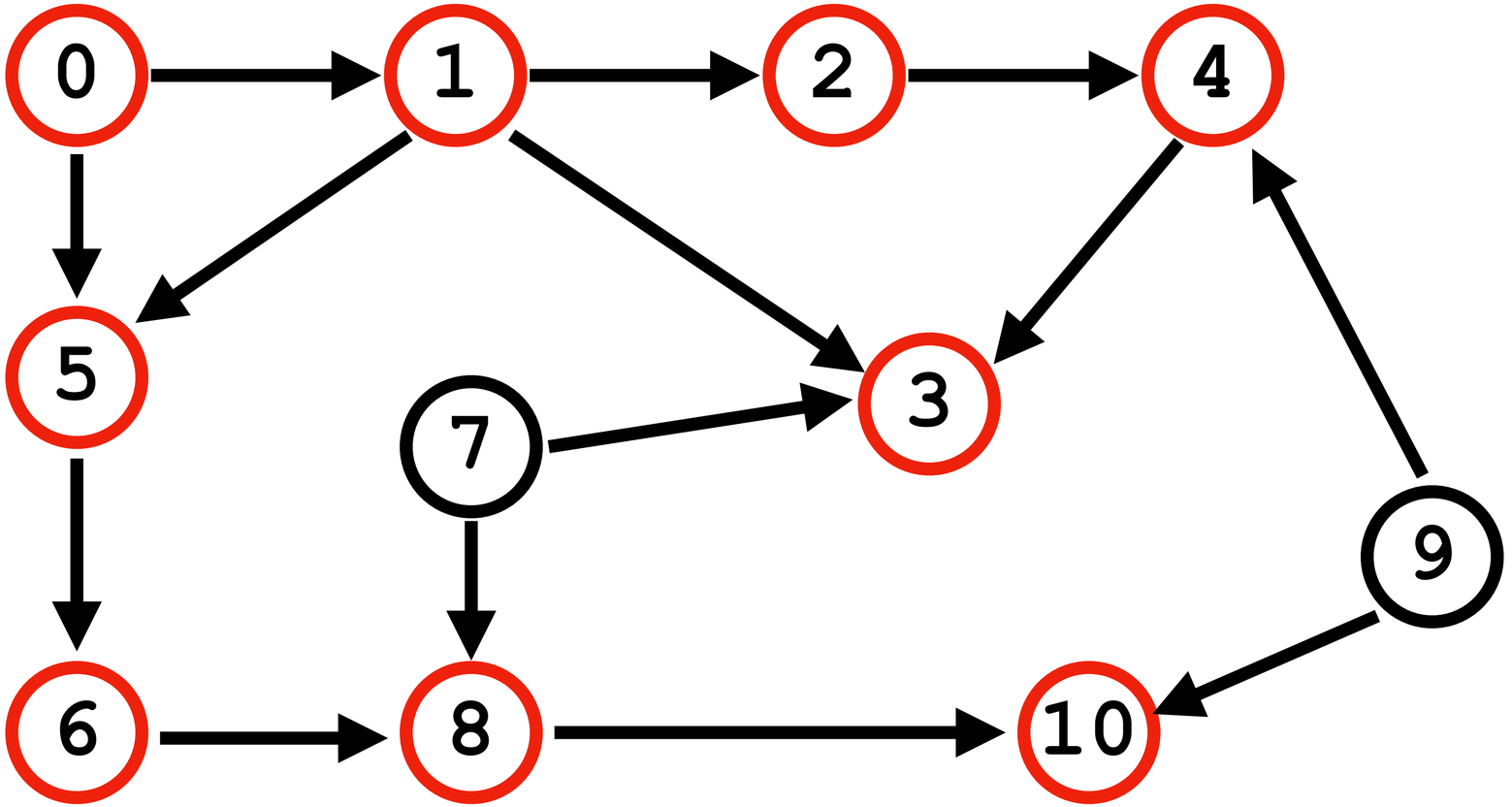
0 1 5 2 3 6 4

BFS Tracing



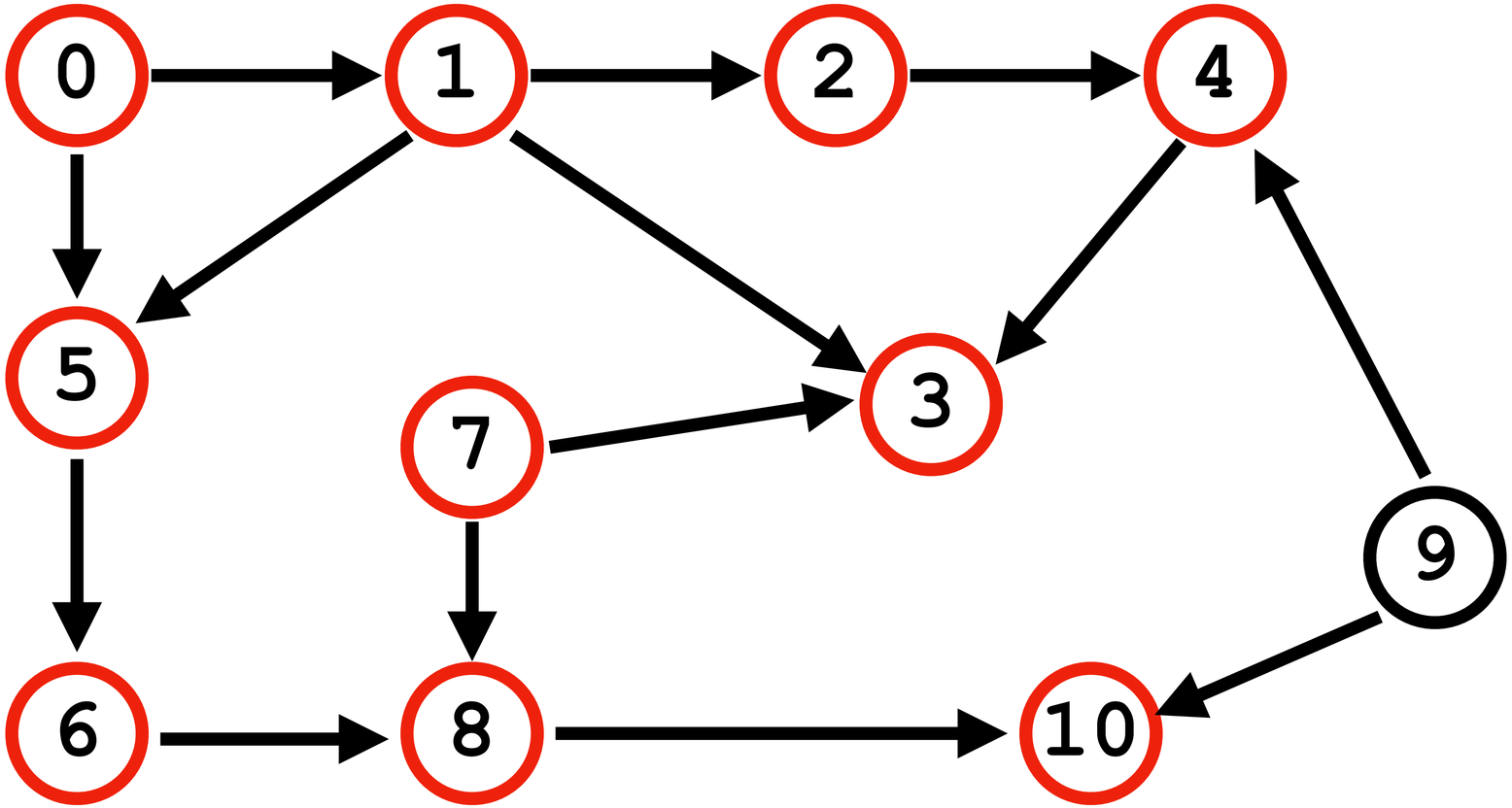
0 1 5 2 3 6 4 8

BFS Tracing



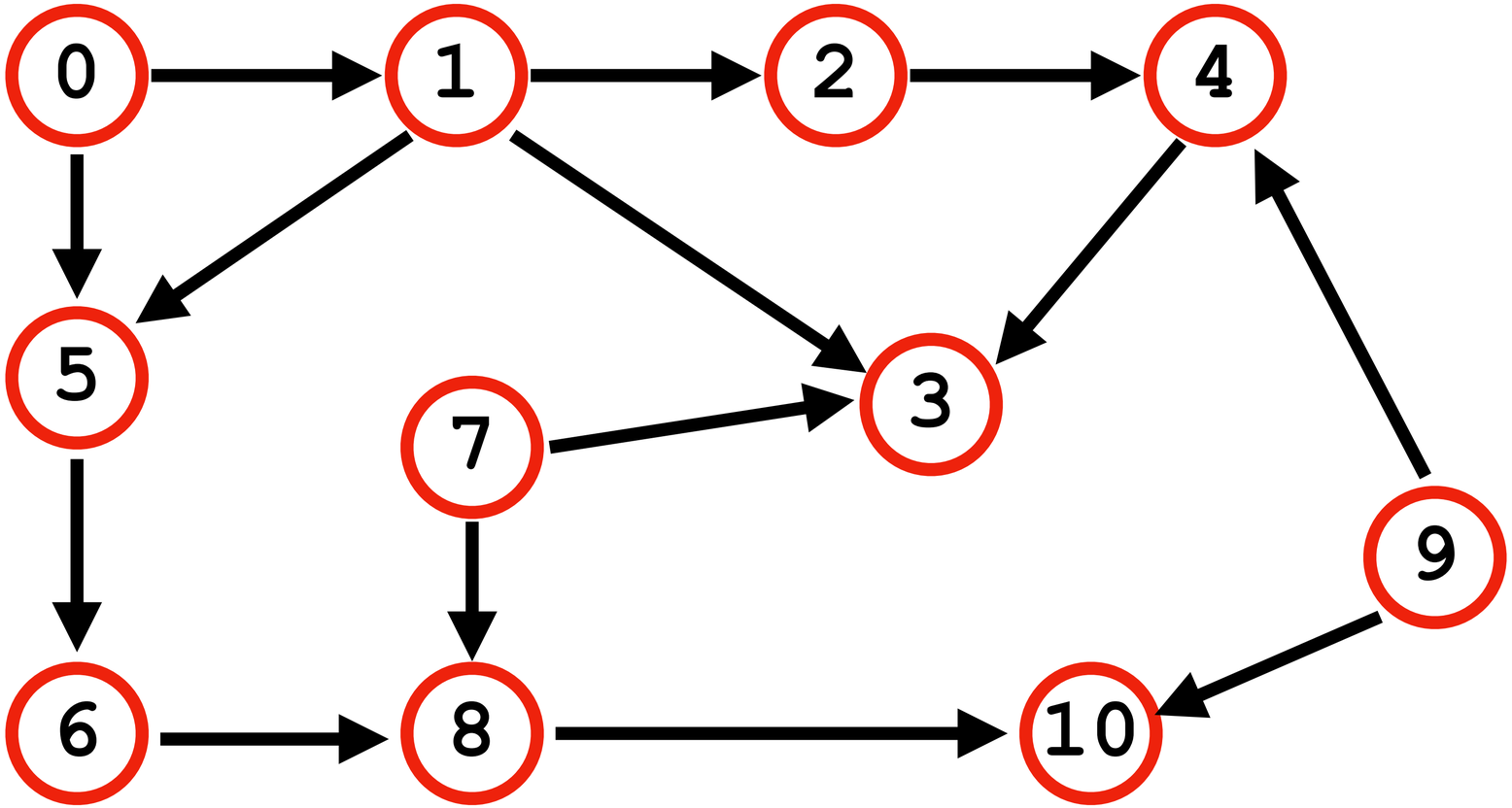
0 1 5 2 3 6 4 8 10

BFS Tracing



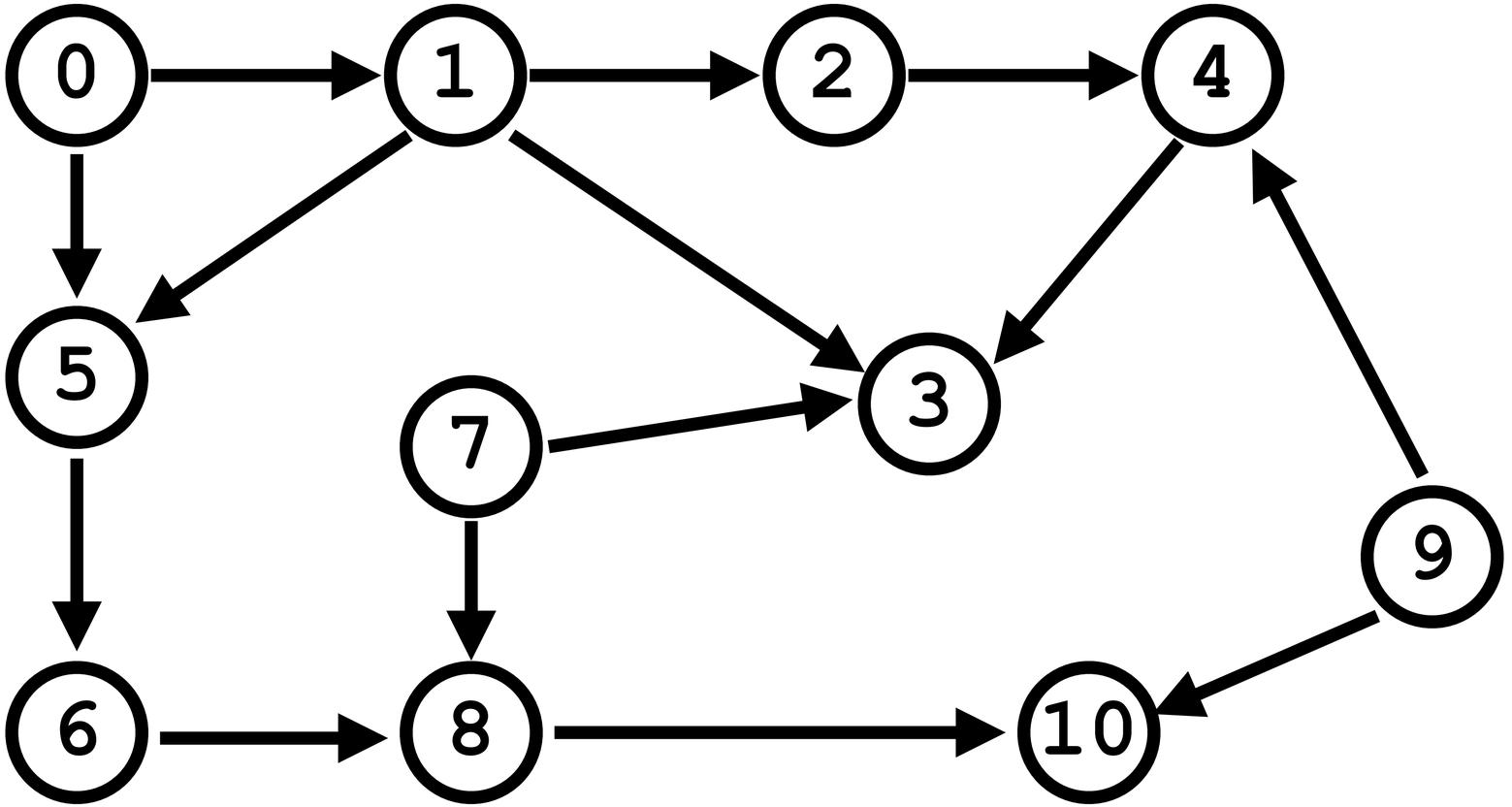
0 1 5 2 3 6 4 8 10 7

BFS Tracing



0 1 5 2 3 6 4 8 10 7 9

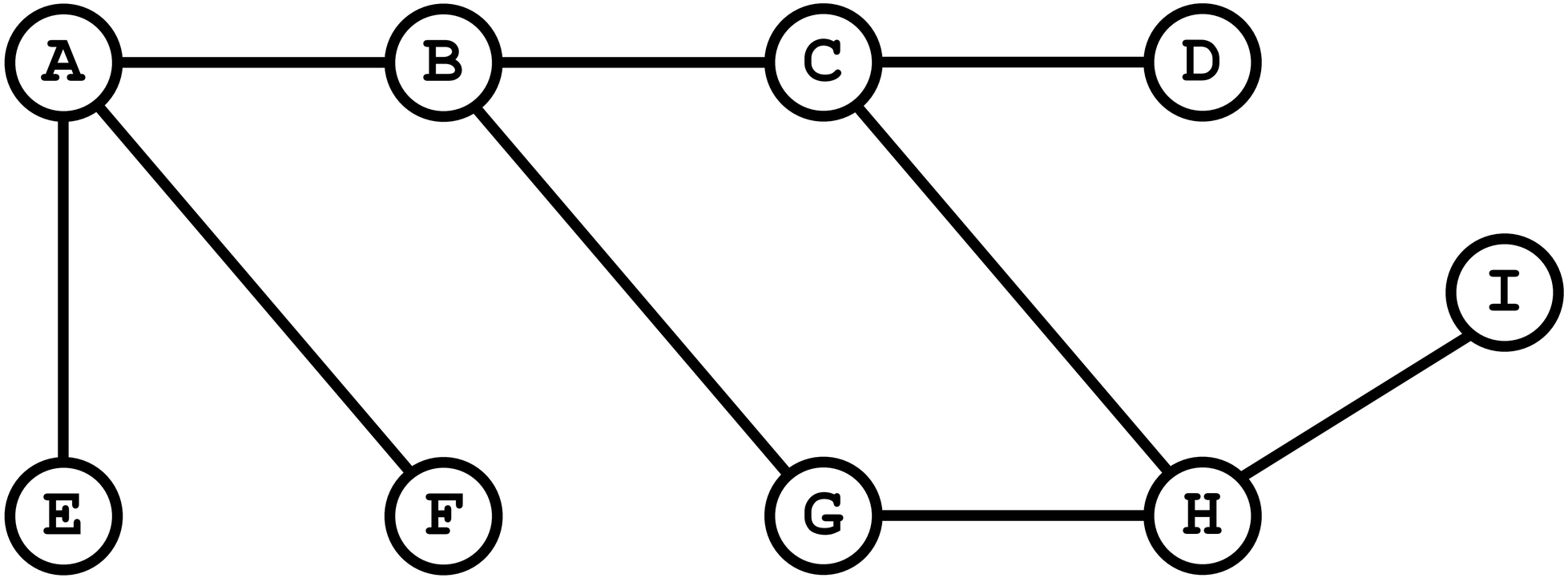
BFS Tracing - Completed Traversal



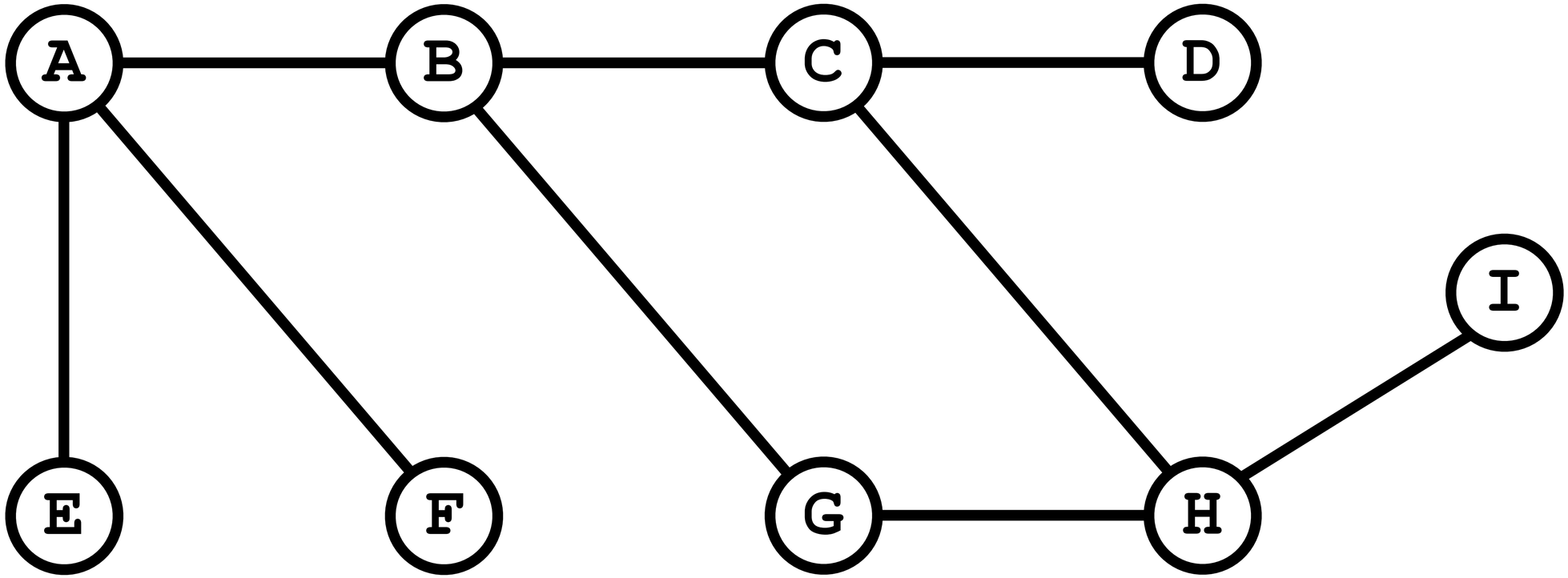
0 1 5 2 3 6 4 8 10 7 9

BFS Tracing - Practice

Perform BFS tracing on the graph below starting from vertex `A`.

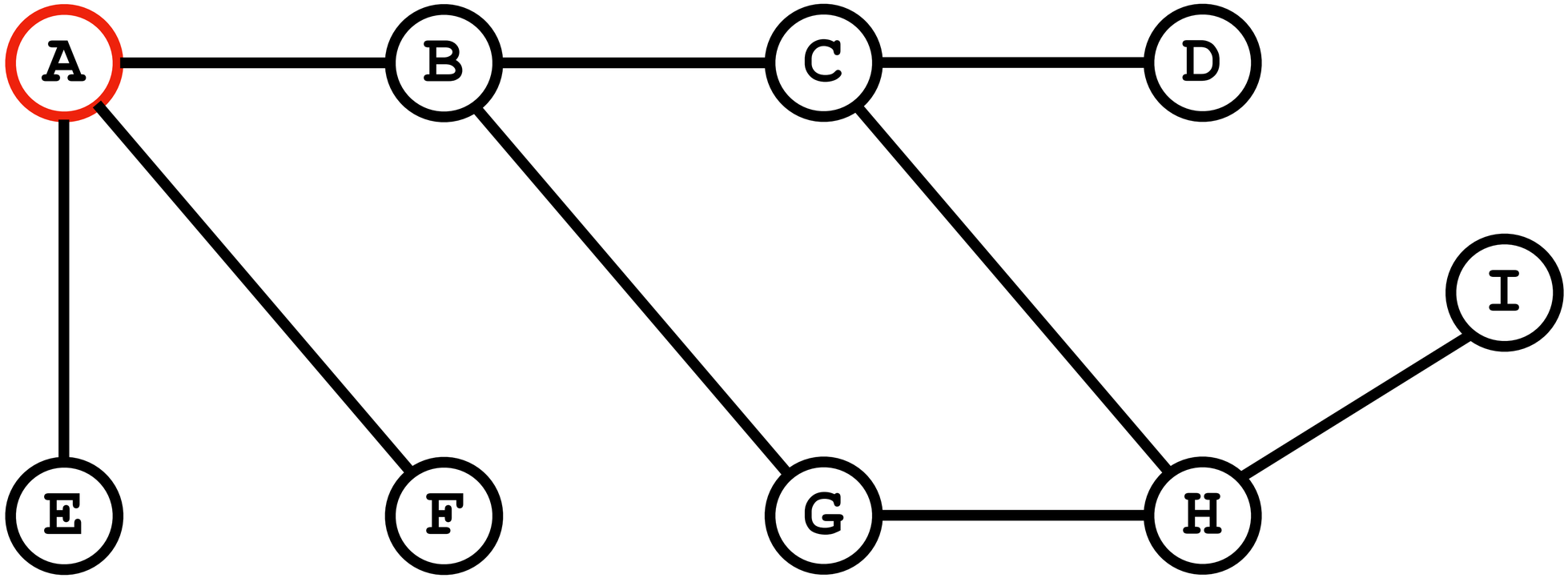


BFS Tracing - Practice (Answer)



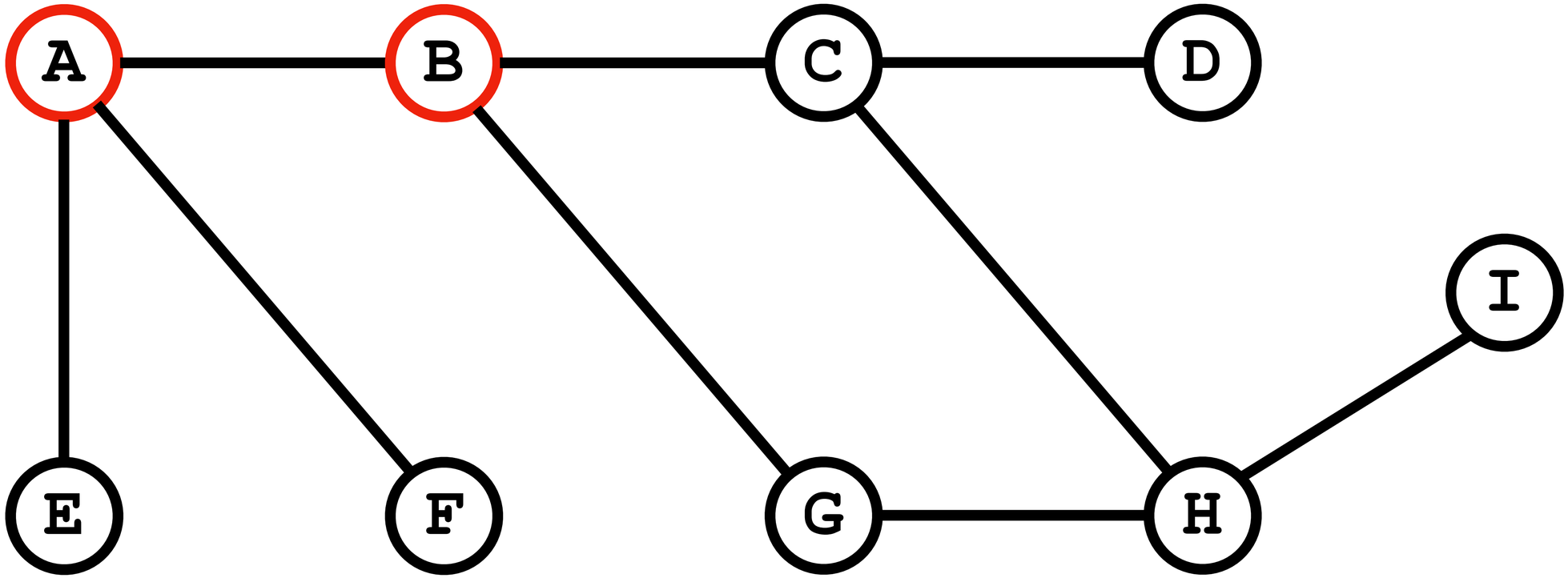
A B E F C G D H I

BFS Tracing - Practice



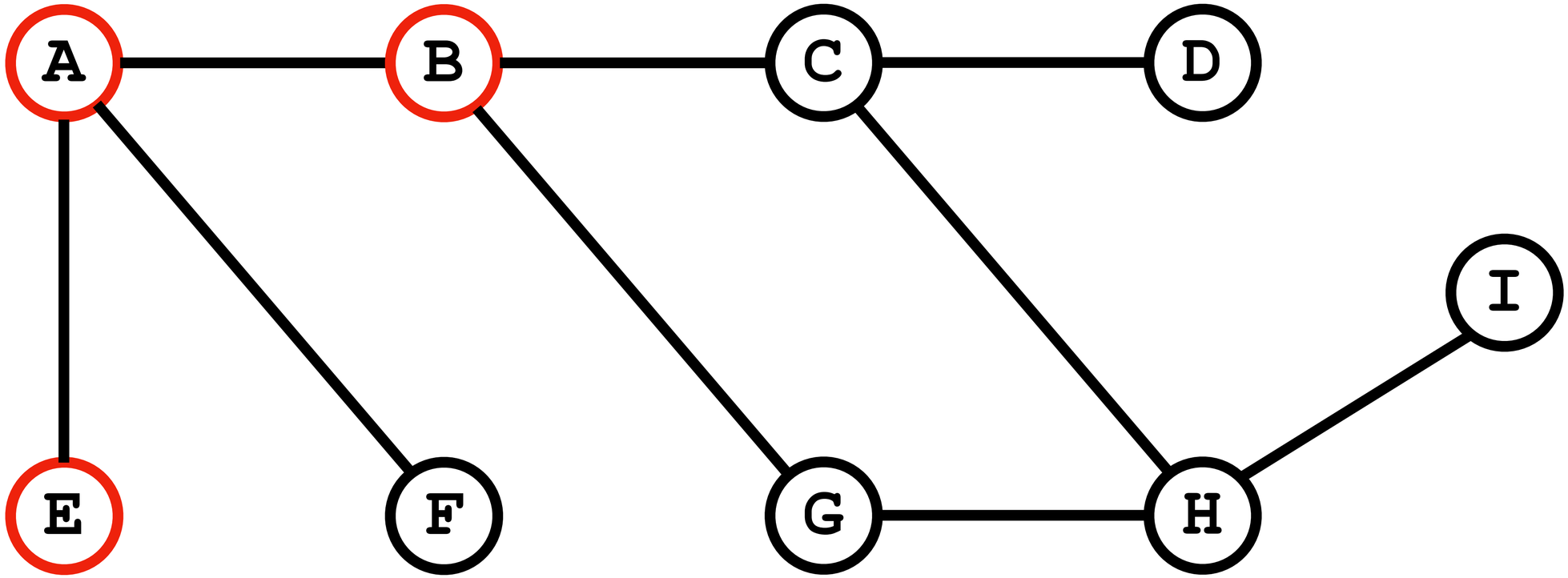
A

BFS Tracing - Practice



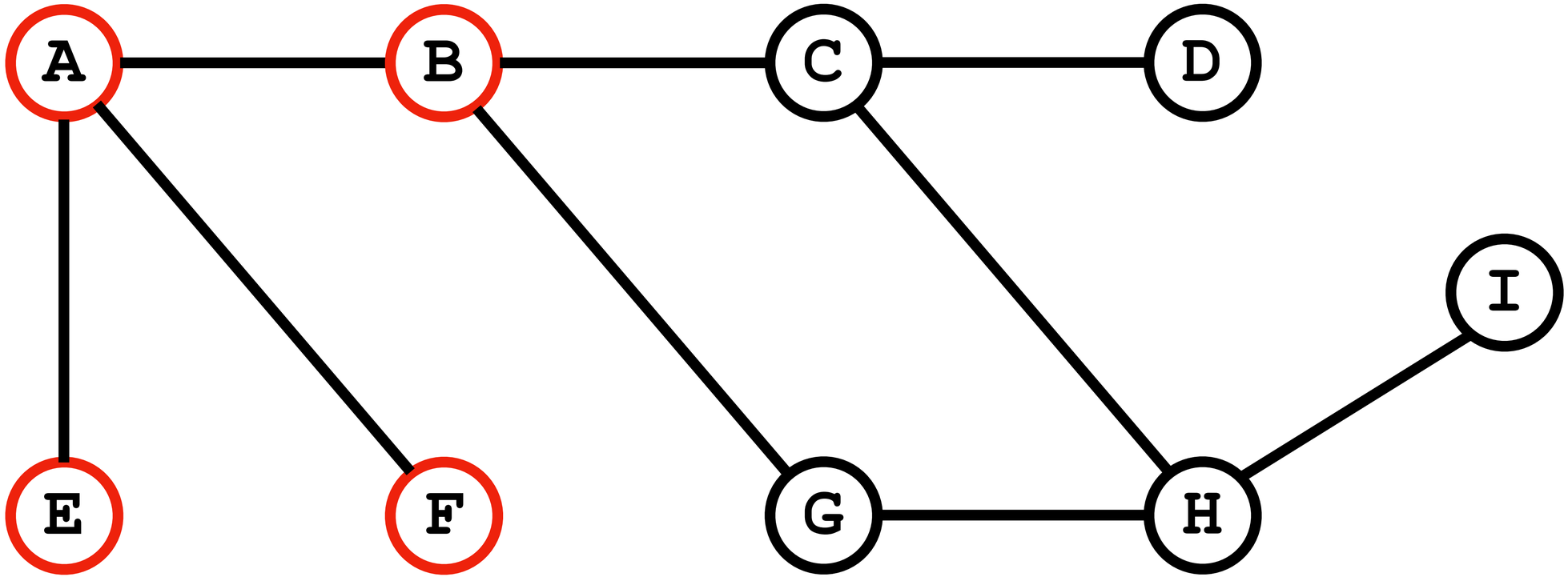
A B

BFS Tracing - Practice



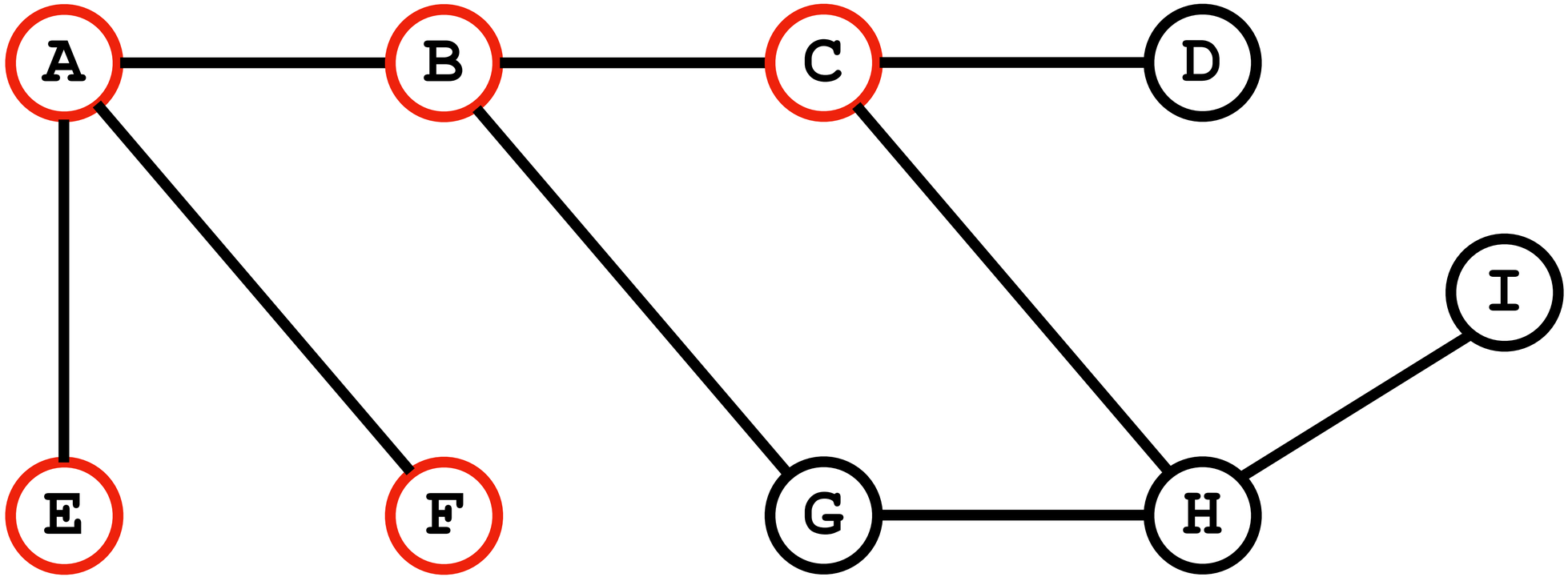
A B E

BFS Tracing - Practice



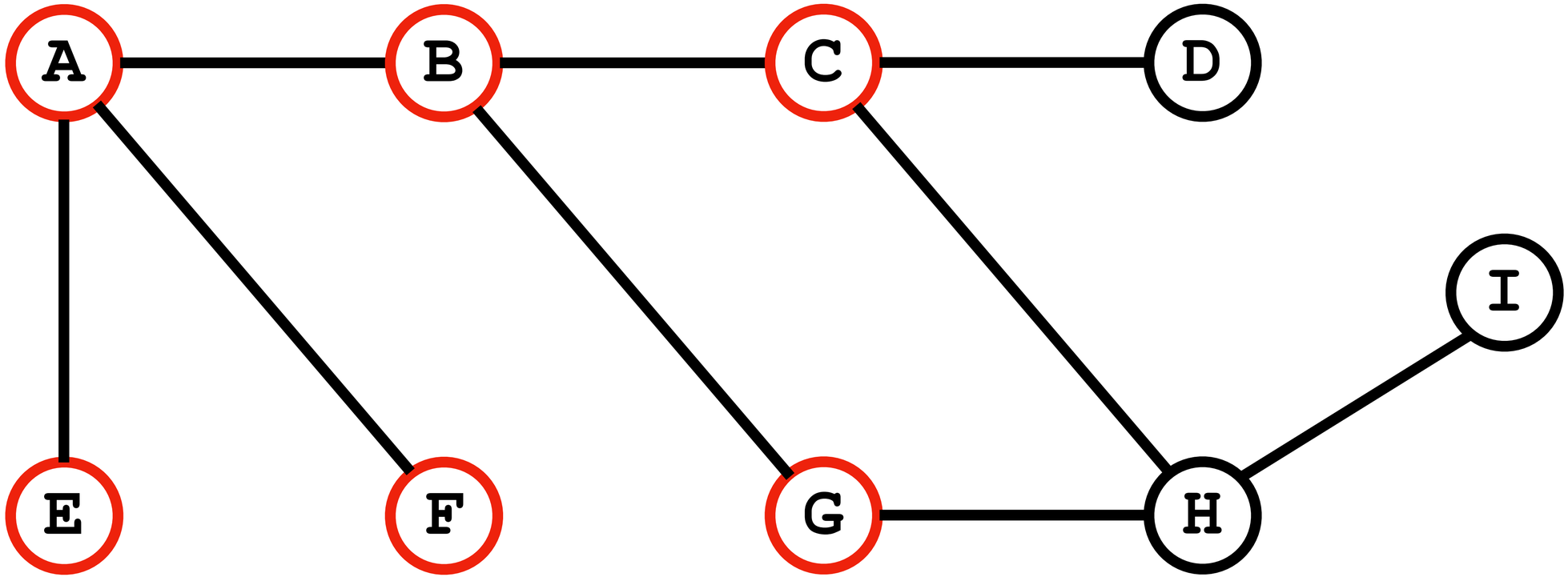
A B E F

BFS Tracing - Practice



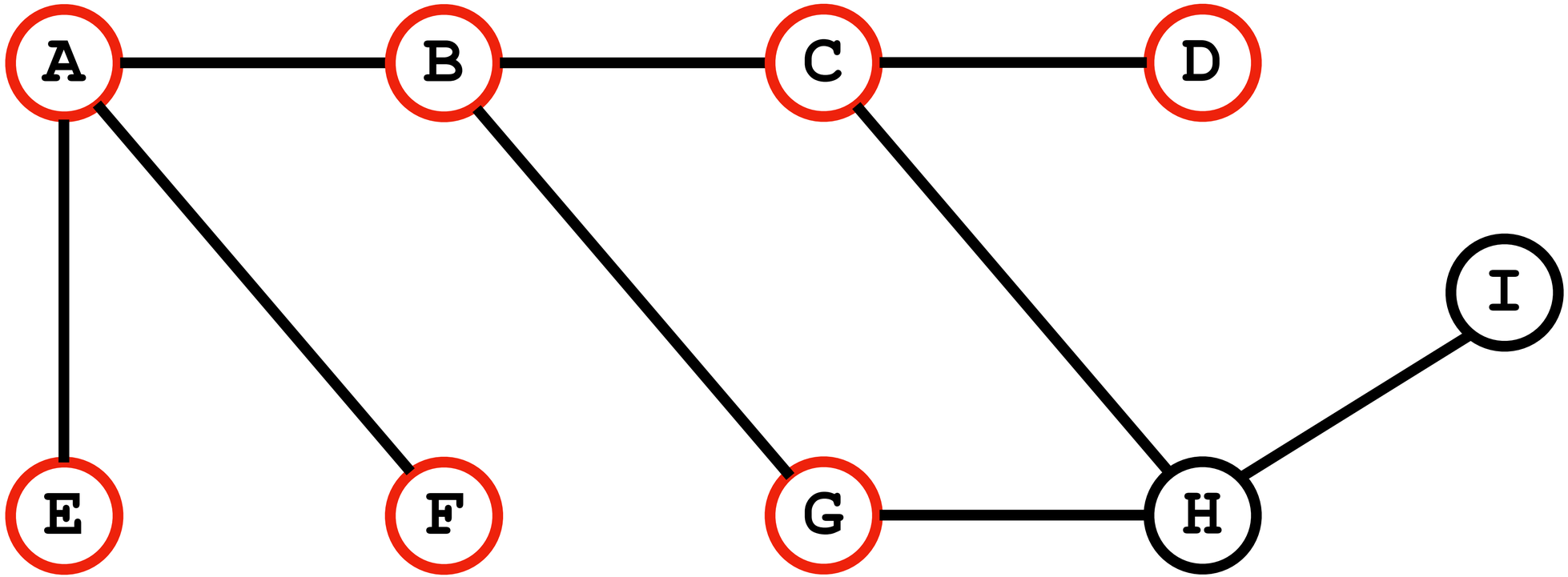
A B E F C

BFS Tracing - Practice



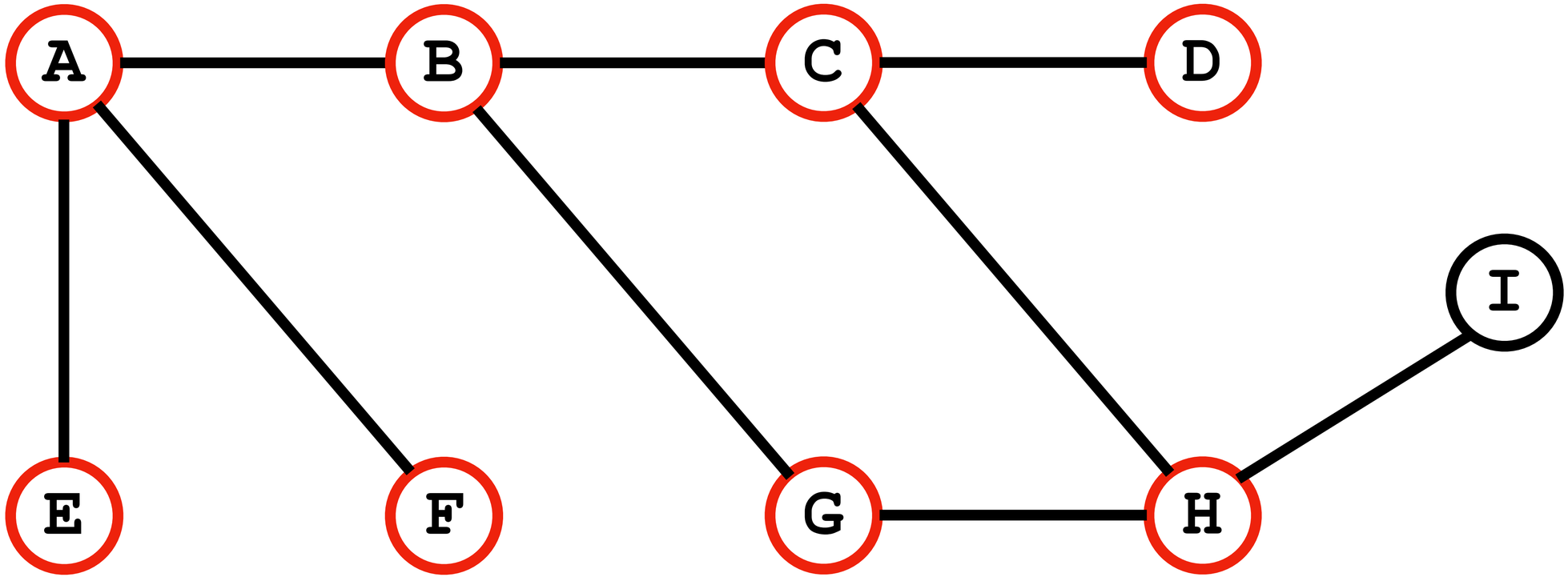
A B E F C G

BFS Tracing - Practice



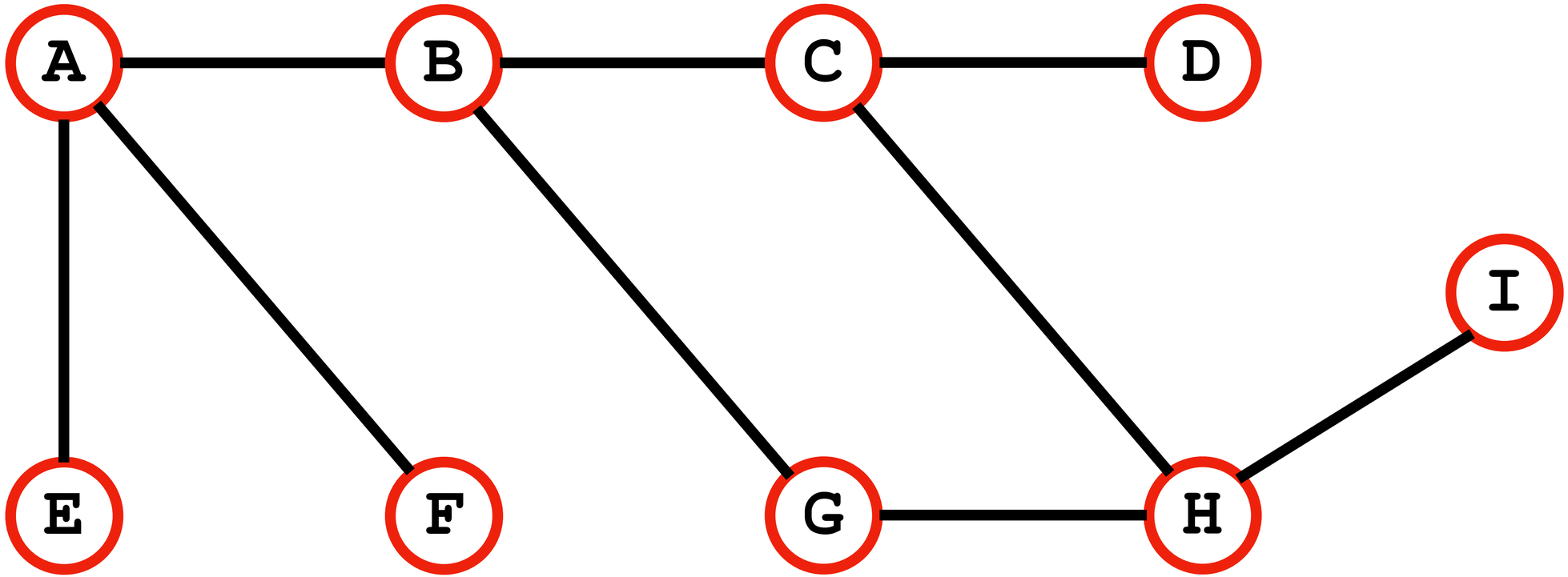
A B E F C G D

BFS Tracing - Practice



A B E F C G D H

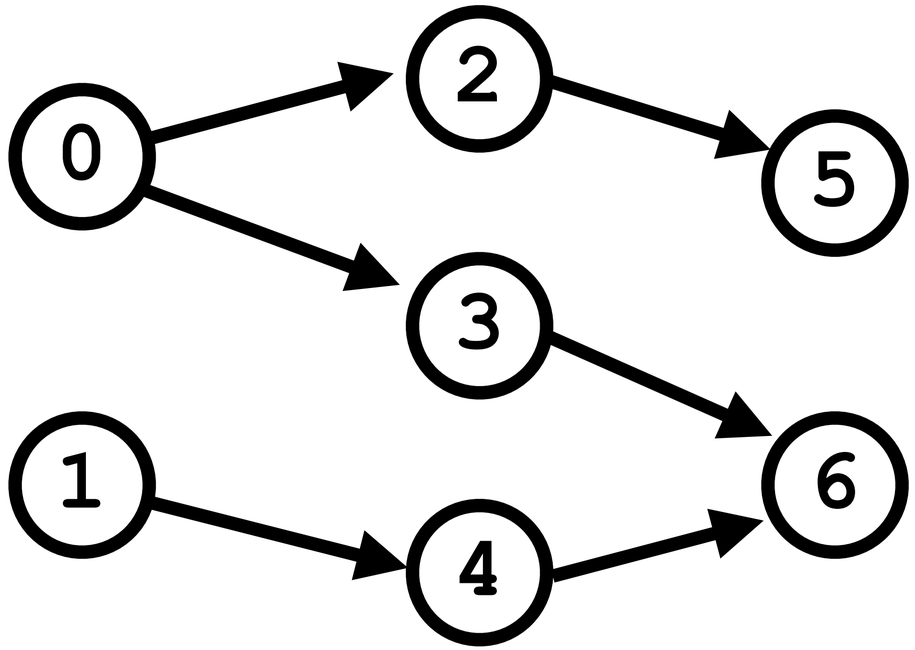
BFS Tracing - Practice



A B E F C G D H I

Extra

The code we've gone over today has shown how to perform DFS and BFS starting from a given source vertex. Implement DFS and BFS using either an adjacency matrix or adjacency list for the entire graph (similar to what we did in the tracing). You can use the graph below in your code.



DFS: 0, 2, 5, 3, 6, 1, 4

BFS: 0, 2, 3, 5, 6, 1, 4