OntarioTech Science

Kourosh Davoudi
heidar.davoudi@ontariotechu.ca

Lecture 9: Graph Foundations

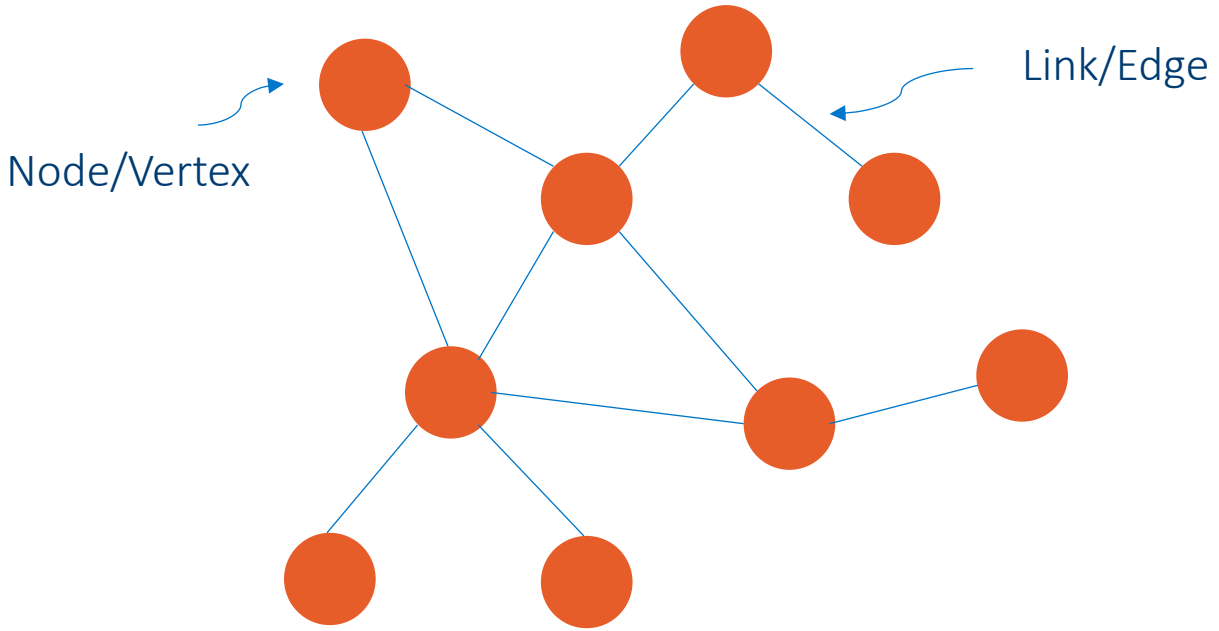
CSCI 3070U: Design and Analysis of Algorithms

Learning Outcomes

- Basic Graph Concepts/Definition
- Graph Representation
- Graph Search
  - Breadth First Search (BFS)
  - Depth First Search (DFS)

What is graph?

- A graph is a collection of nodes (vertices) and links (edges) between them.



Why Graphs?

Graphs are used to solve many real-world problems:

Social Graph

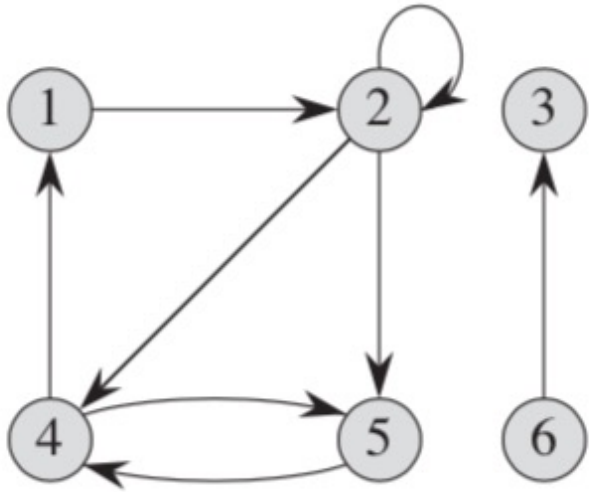
Transportation Graph

Graph Definitions

- Directed Graph:
  - A directed graph (or digraph) G is a pair (V, E), where V is a finite set of vertices and set of edges E is a binary relation on V
  - The edges are directional: $(v_1, v_2) \neq (v_2, v_1)$
  - Example: a directed graph $G = (V, E)$, where

$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{ (1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)\}$$

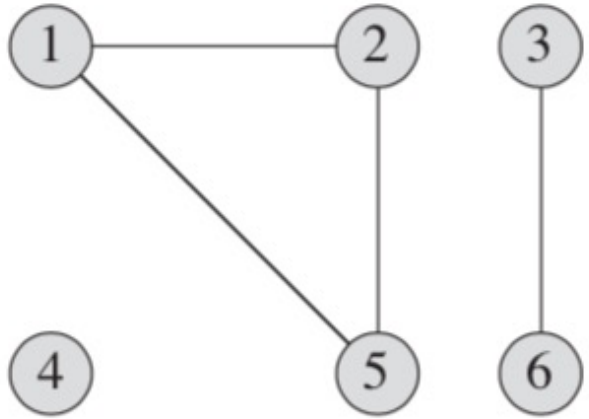


Graph Definitions

- Undirected Graph
  - An undirected graph is $G = (V, E)$, where V is a finite set vertices and the edge set E consisting of unordered pairs of vertices, rather than ordered pairs.
  - The edges are not directional: $\{v_1, v_2\} = \{v_2, v_1\}$
  - Example: a (undirected) graph $G = (V, E)$, where

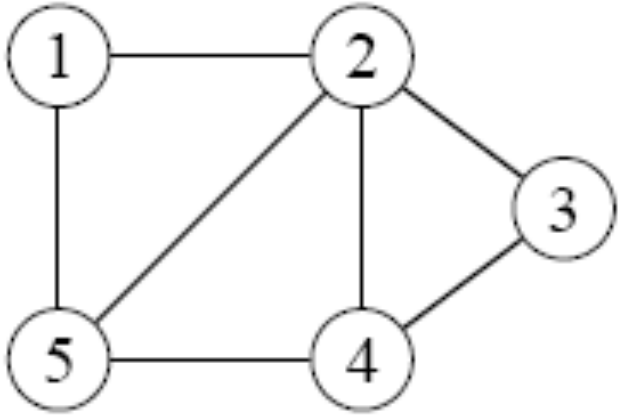
$$V = \{1, 2, 3, 4, 5, 6\}$$

$$E = \{ \{1,2\}, \{1,5\}, \{2,5\}, \{3,6\}\}$$



Representation of Graphs

- Adjacency Matrix

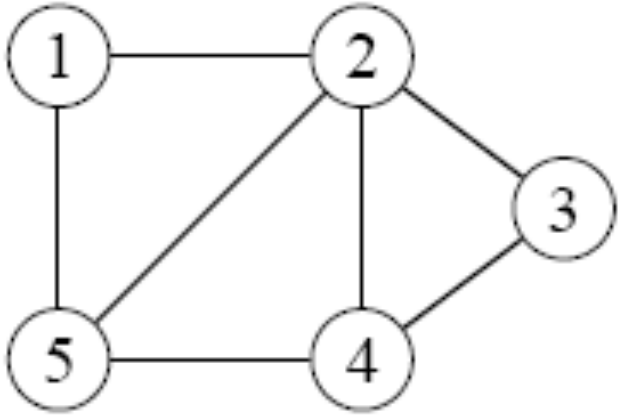


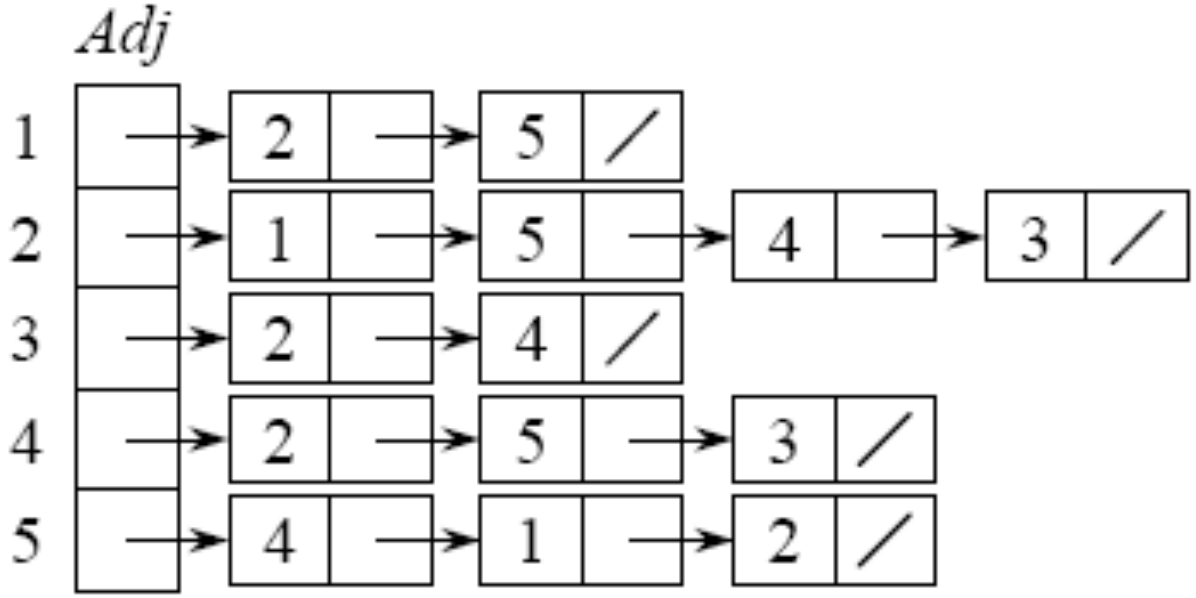
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

$$a_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \text{ ,} \\ 0 & \text{otherwise .} \end{cases}$$

Representation of Graphs

- Adjacency List





Representation of Graphs

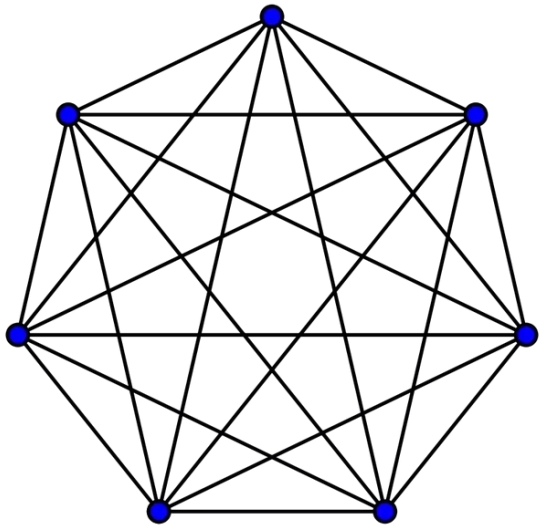
	Adjacency Matrix	Adjacency List
Time		
• If $(u, v) \in E$	$\Theta(1)$	$O(deg(u))$
• To list all adjacent vertices adjacent to u	$\Theta(V)$	$\Theta(deg(u))$
Space		
• Space complexity	$\Theta(V^2)$	$\Theta(V + E)$

Representation of Graphs

Adjacency Matrix

Appropriate for Dense Graph

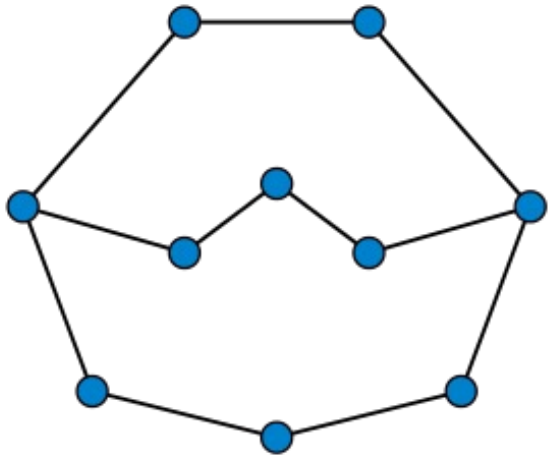
$$E \approx V^2$$



Adjacency List

Appropriate for Sparse Graph

$$E \ll V^2$$



Graph Properties/Concepts

- Path
  - The path form v_i to v_j contains the vertices : v_i, v_{i+1}, ..., v_{j-1}, v_j such that (v_i, v_{i+1}), ..., $(v_{j-1}, v_j) \in E$

$$\text{Path} : v_i \rightarrow v_{i+1} \rightarrow \ \ldots \rightarrow \ v_{j-1} \rightarrow \ v_j$$

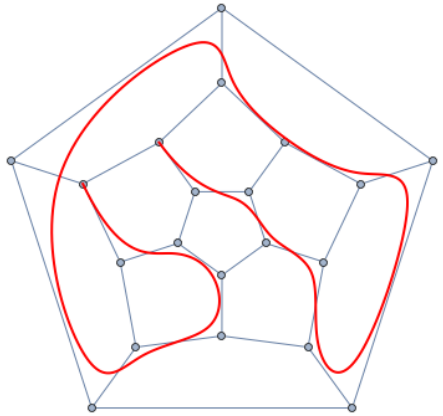
- Cycle
  - A cycle is a (non-zero) sequence of vertices, each connected by an edge, ending at the same vertex

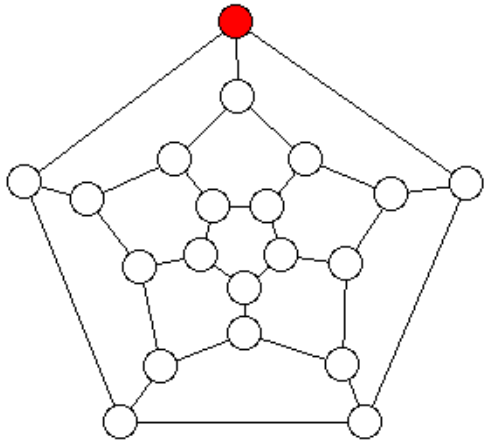
$$\text{Cycle}: v_i \rightarrow v_{i+1} \rightarrow \ \ldots \rightarrow \ v_{j-1} \rightarrow \ v_i$$

A graph is acyclic if it has no cycles

Graph Properties/Concepts

- Hamiltonian cycle:
  - A cycle which visits every vertex exactly once, ending at the same vertex

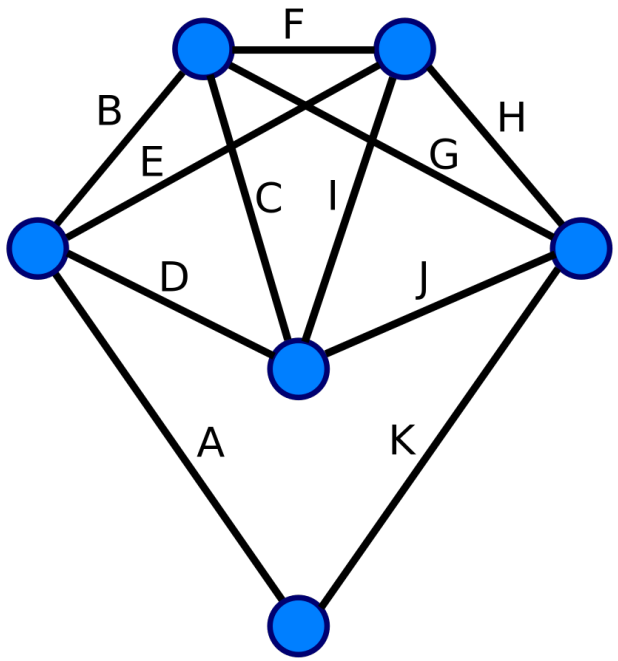




  - The travelling salesperson problem (TSP) is really the problem of finding a Hamiltonian cycle with minimum total cost in an arbitrary graph

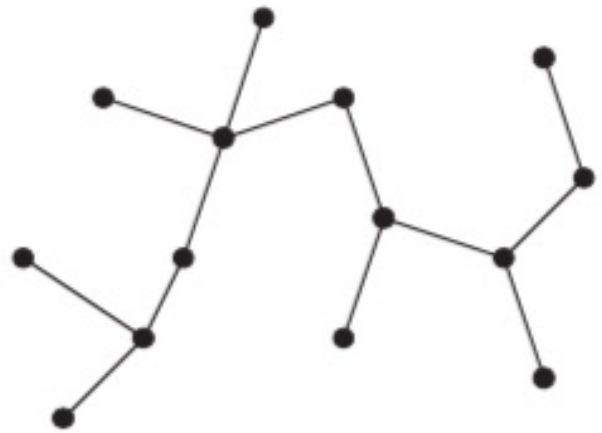
Graph Properties/Concepts

- Euler tour:
  - A cycle which traverses every edge exactly once, but can visit the same vertex multiple times



Graph Properties/Concepts

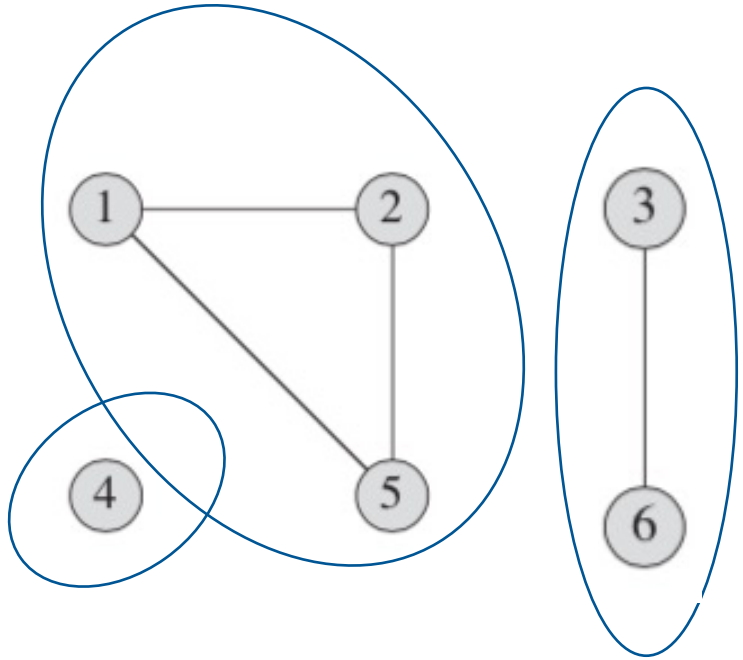
- Trees
  - are a special case of graphs
    - Connected and acyclic
    - Only a single path between any two nodes



- An acyclic graph that is not connected is called a forest

Graph Properties/Concepts

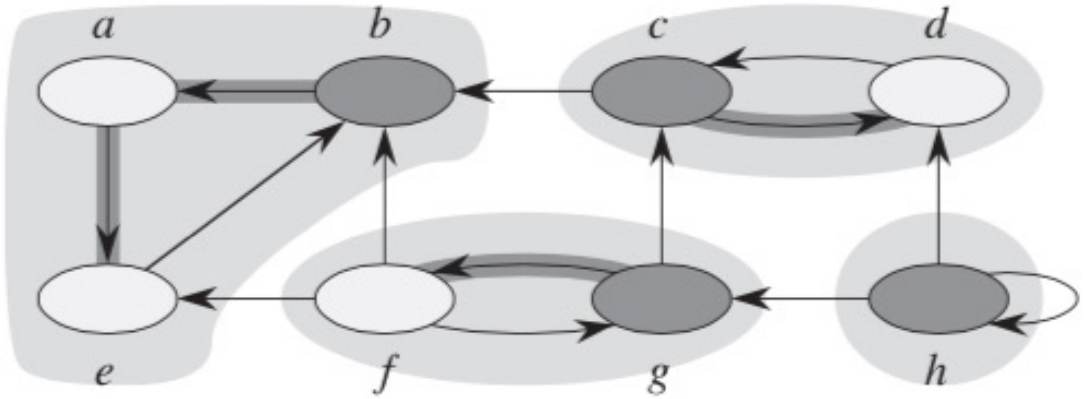
- Connected Components
  - A maximal set of vertices $C \subseteq V$ in a undirected graph such that every vertex is reachable from others.



  - An undirected graph is connected if it has only one connected component.

Graph Properties/Concepts

- Strongly Connected Components :
  - A maximal set of vertices $C \subseteq V$ in a directed graph such that every pair of vertices u and v in C , both are reachable from each other.

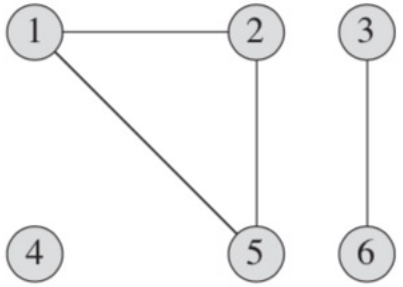


  - A directed graph is strongly connected if it has only one strongly connected component.

Graph Properties/Concepts

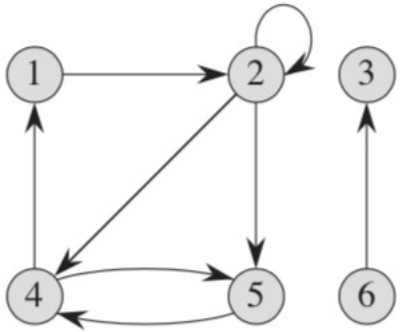
- Node degree:
  - The degree of a vertex in an undirected graph is the number of edges incident on it.

degree(5) = 2
degree(4) = 0



- Node In-degree/Out-degree:
  - out-degree of a vertex: the number of edges leaving it
  - in-degree of a vertex is the number of edges entering it.
  - degree = in-degree + out-degree

in-degree (2) = 2
out-degree(2) = 3



Graph Search

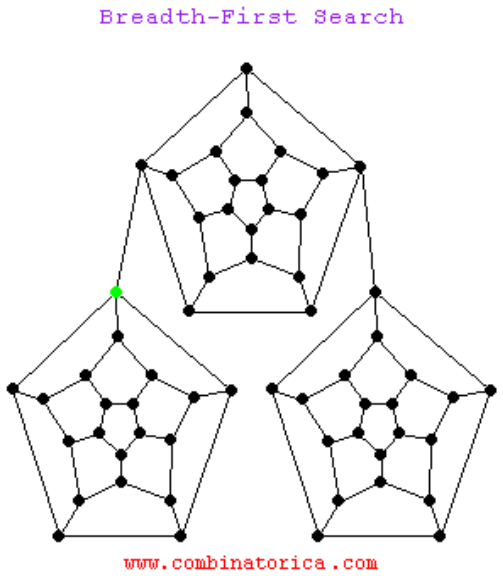
- What is graph search?
  - It is a general technique for traversing through graphs. Namely, we visit each vertex of the graph without repetition.

  - Example :
    - Sliding puzzle
      - Vertices: state in the puzzle
      - Edge: connect to adjacent state

1	6	5
7	3	8
	4	2

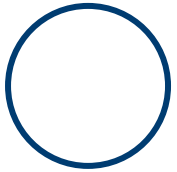
Breadth First Search (BFS)

- Input:
  - Graph G=(V, E), either directed or undirected, and source vertex $s \in V$
- Output:
  - $v.d$ = distance (smallest # of edges) from s to v , for all $v \in V$
  - $v.\pi$ = predecessor of vertex v along the shortest path from source s.

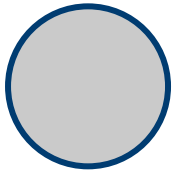


Breadth First Search (BFS)

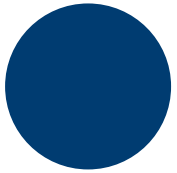
- To keep track of progress, breadth first search colors each vertex white, gray, or black.
  - White vertices have not been discovered
    - All vertices start out white



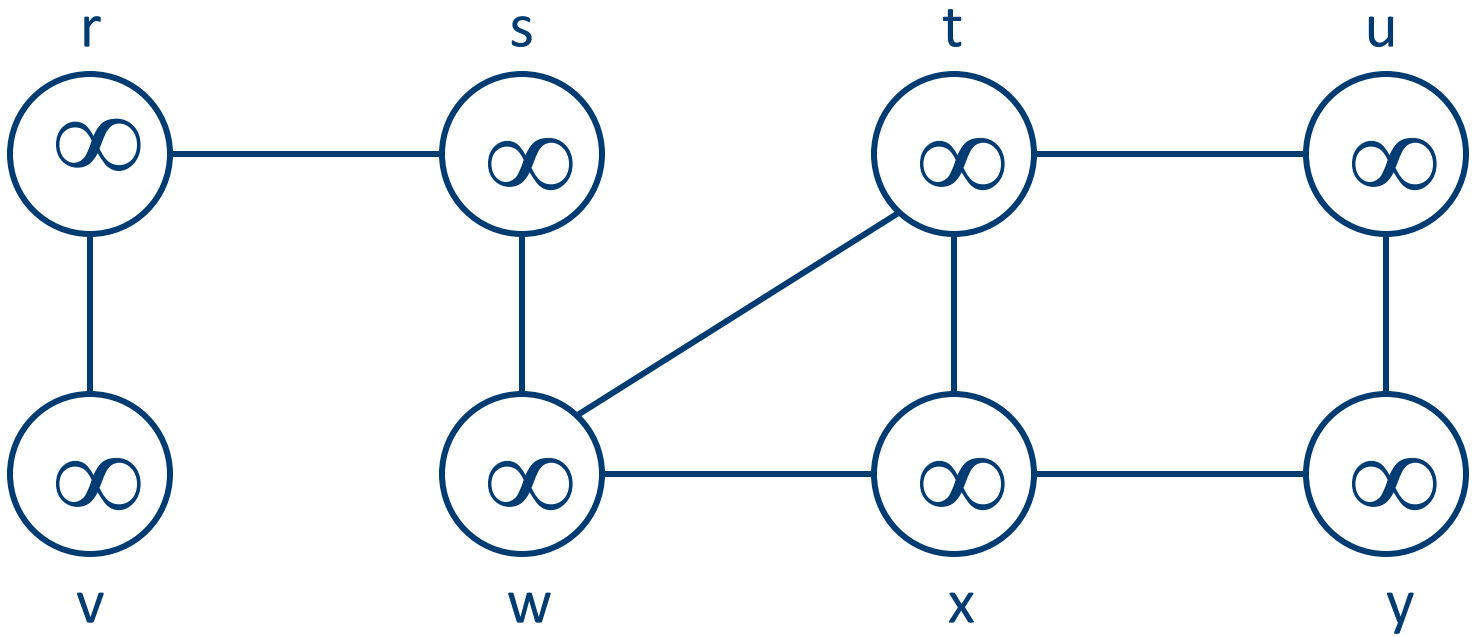
  - Grey vertices are discovered but not fully explored
    - They are visited but waiting for neighborhood visits



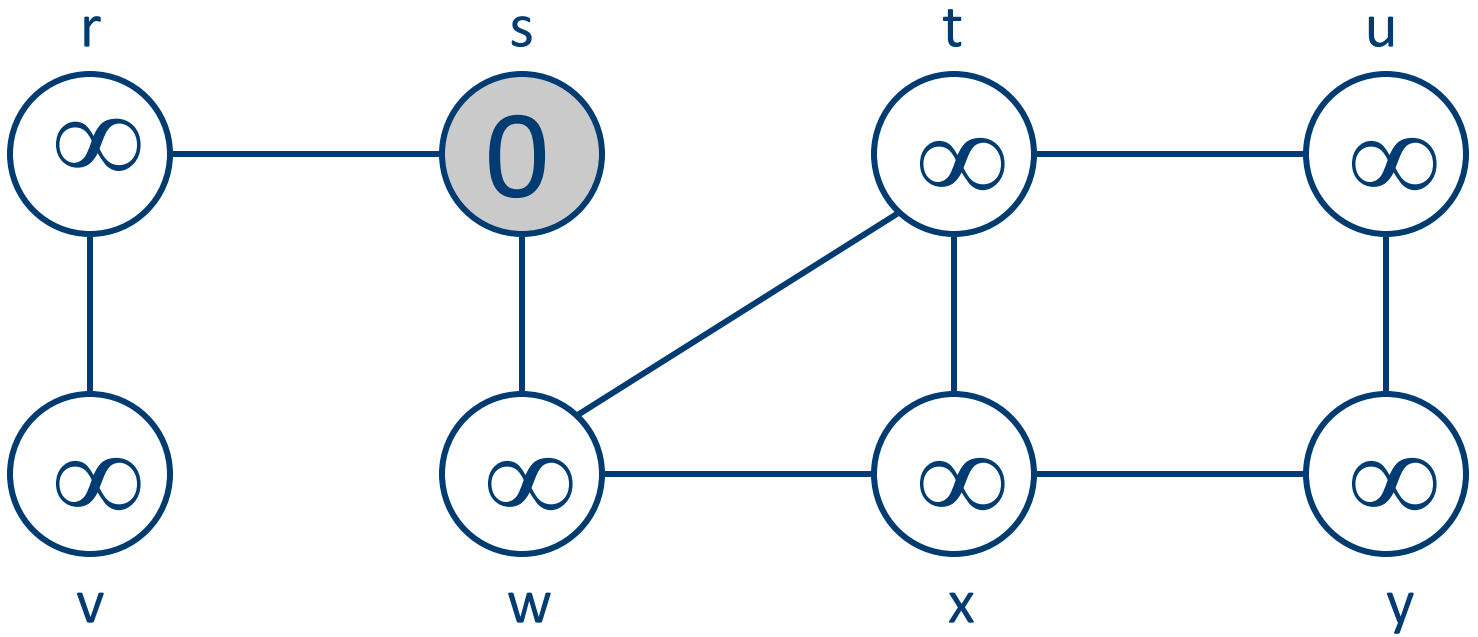
  - Black vertices are discovered and fully explored
    - They are adjacent only to black and gray vertices



BFS: Example

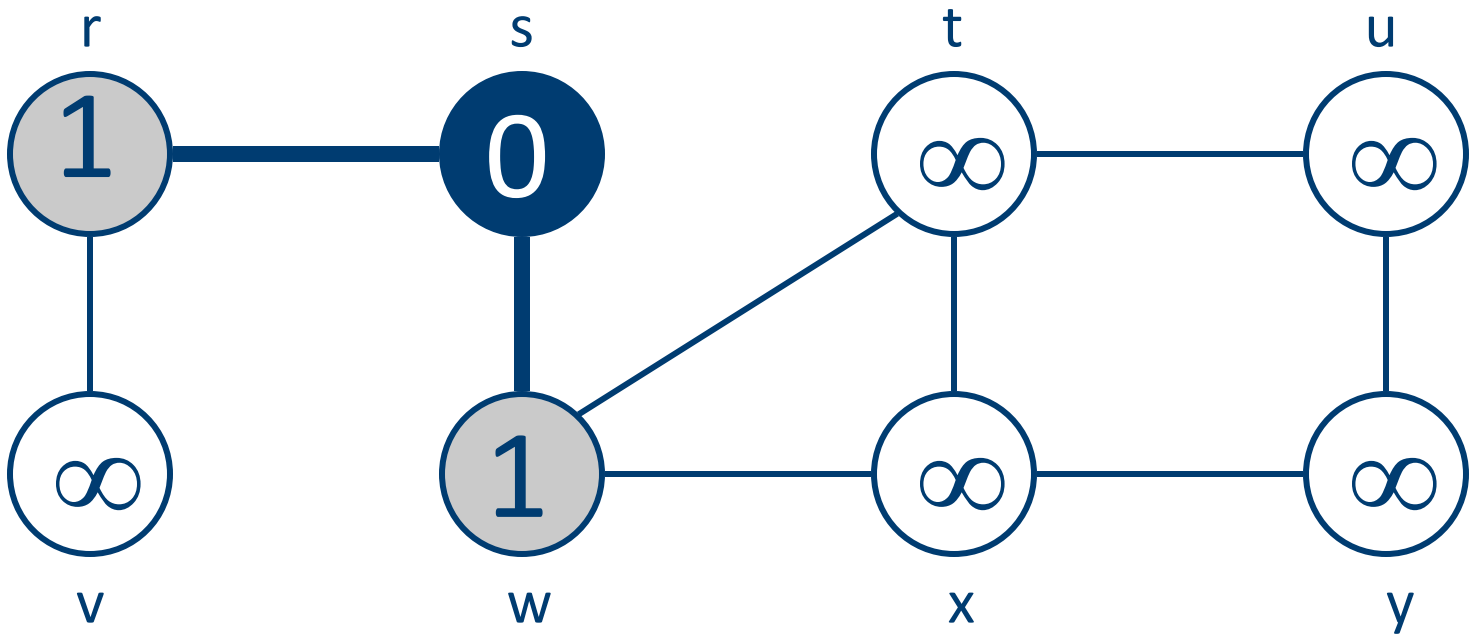


BFS: Example



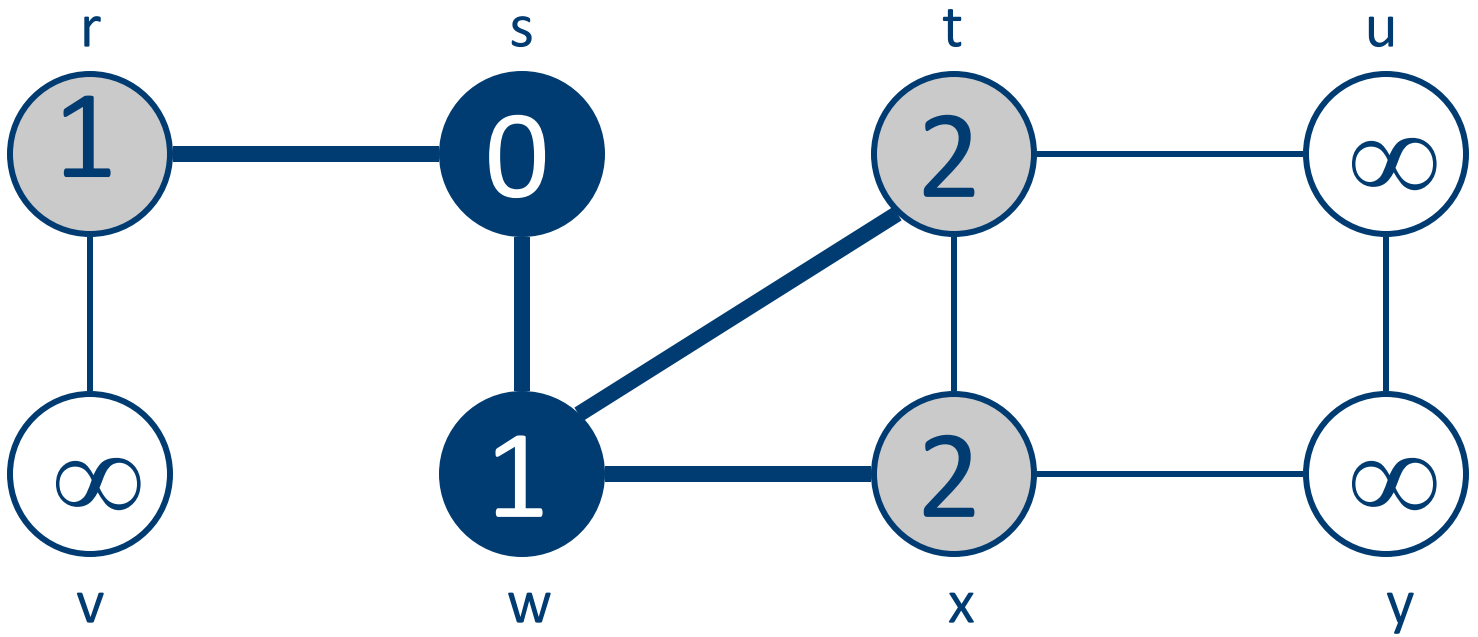
Q: | s |

BFS Example



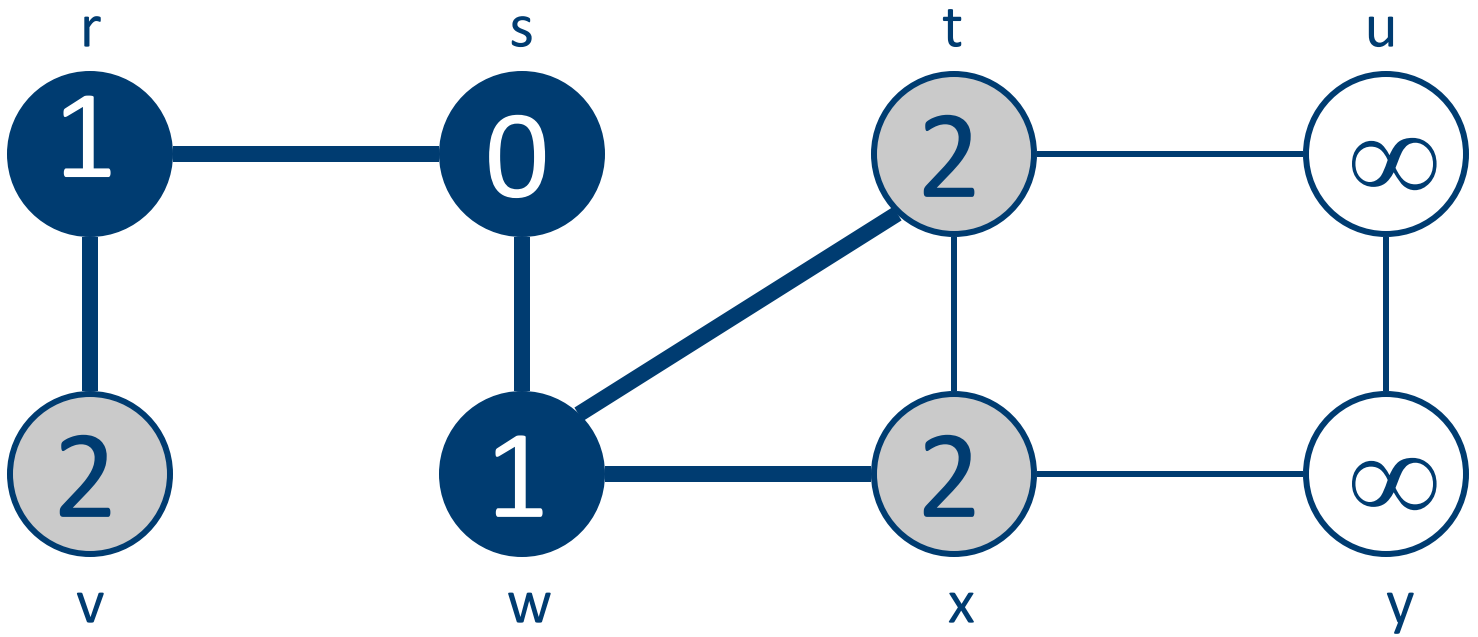
Q: | w | r |

BFS Example



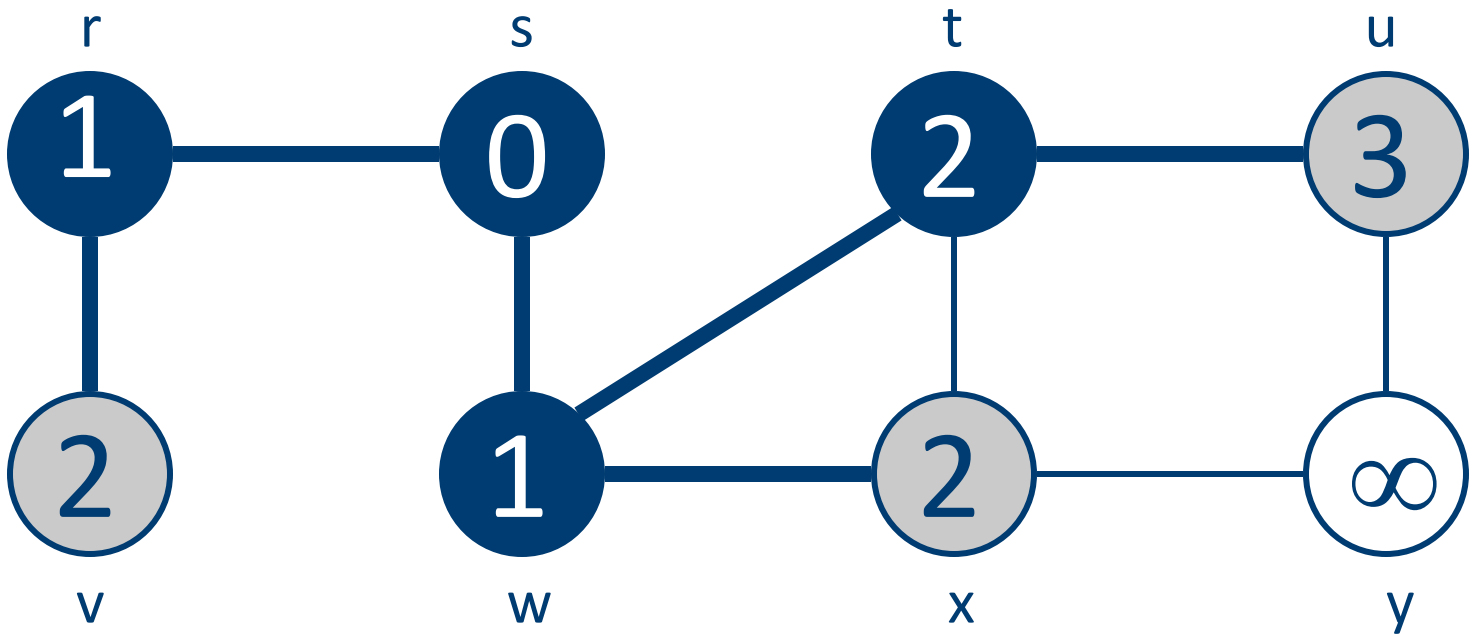
Q: | r | t | x |

BFS Example



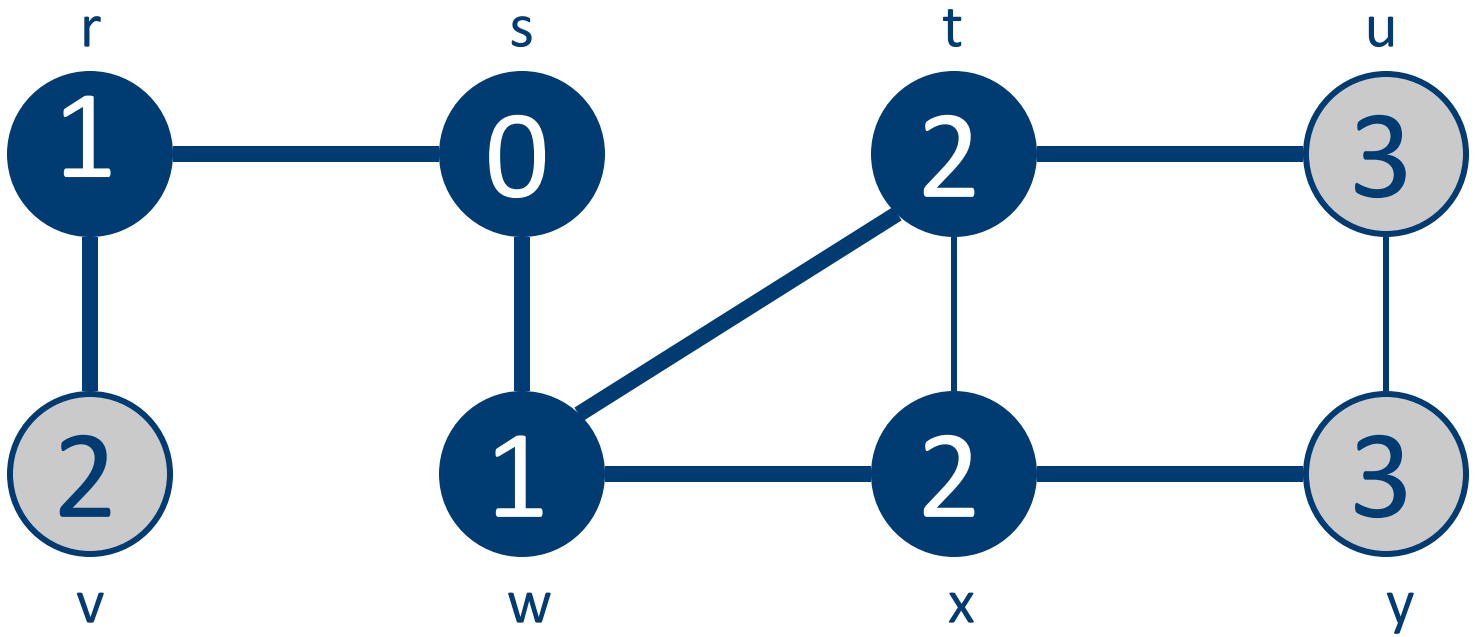
Q: | t | x | v |

BFS Example



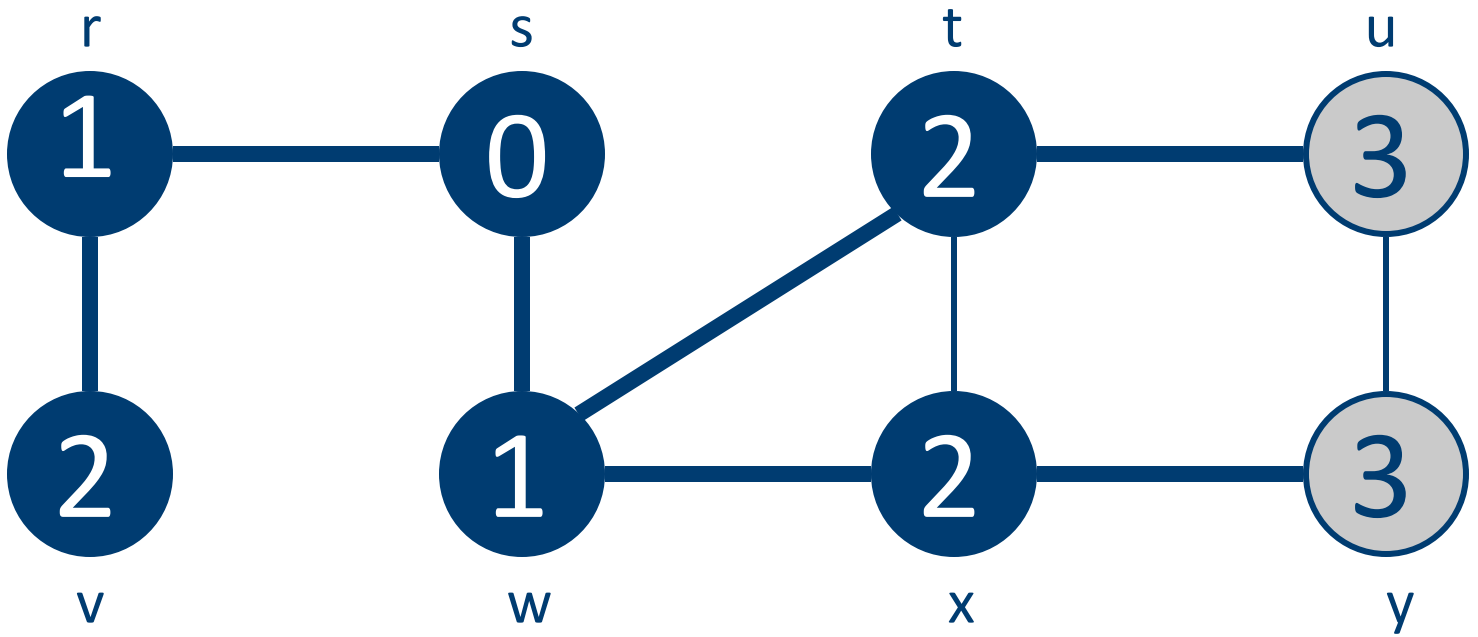
Q: | x | v | u |

BFS Example



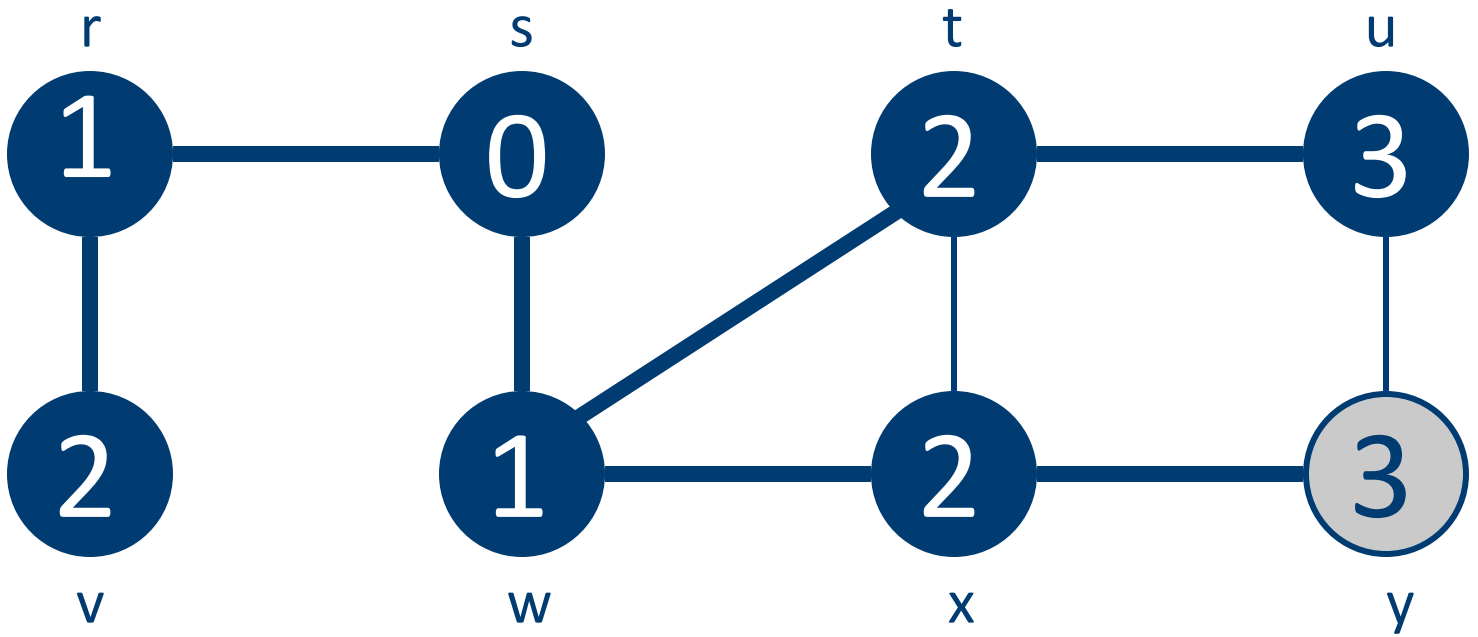
Q: | v | u | y |

BFS Example



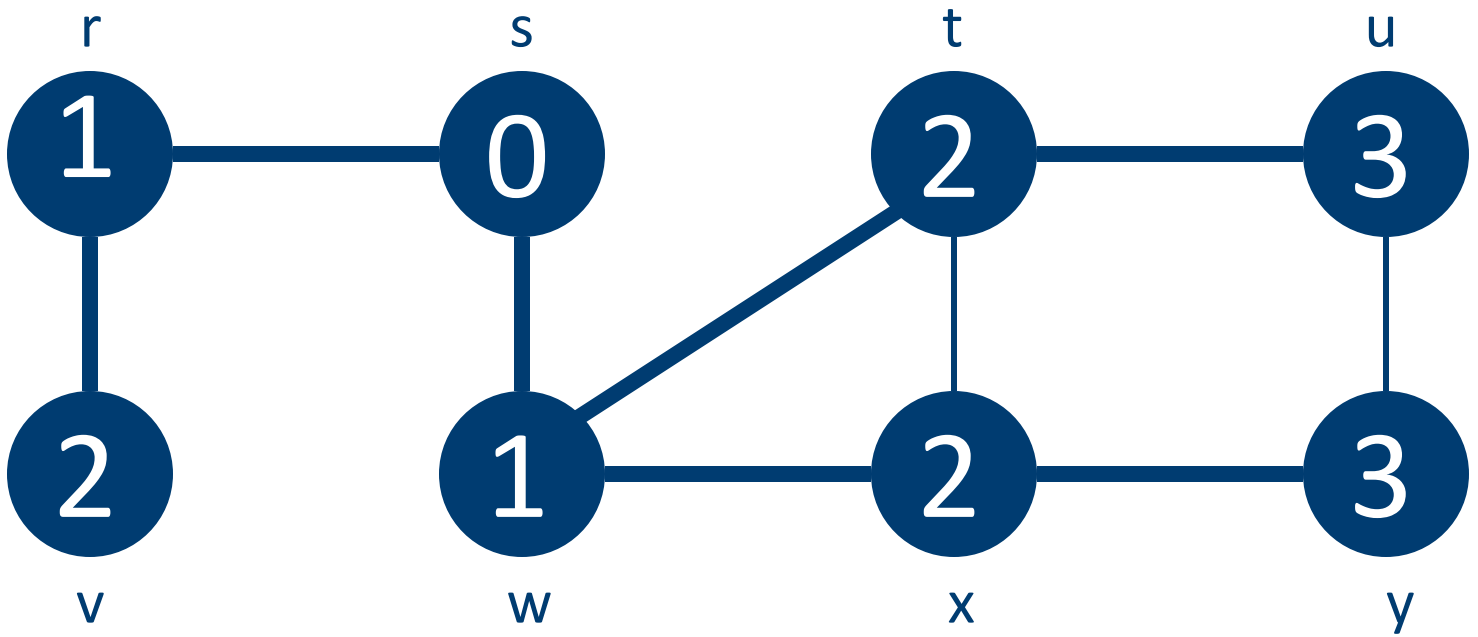
Q: | u | y |

BFS Example



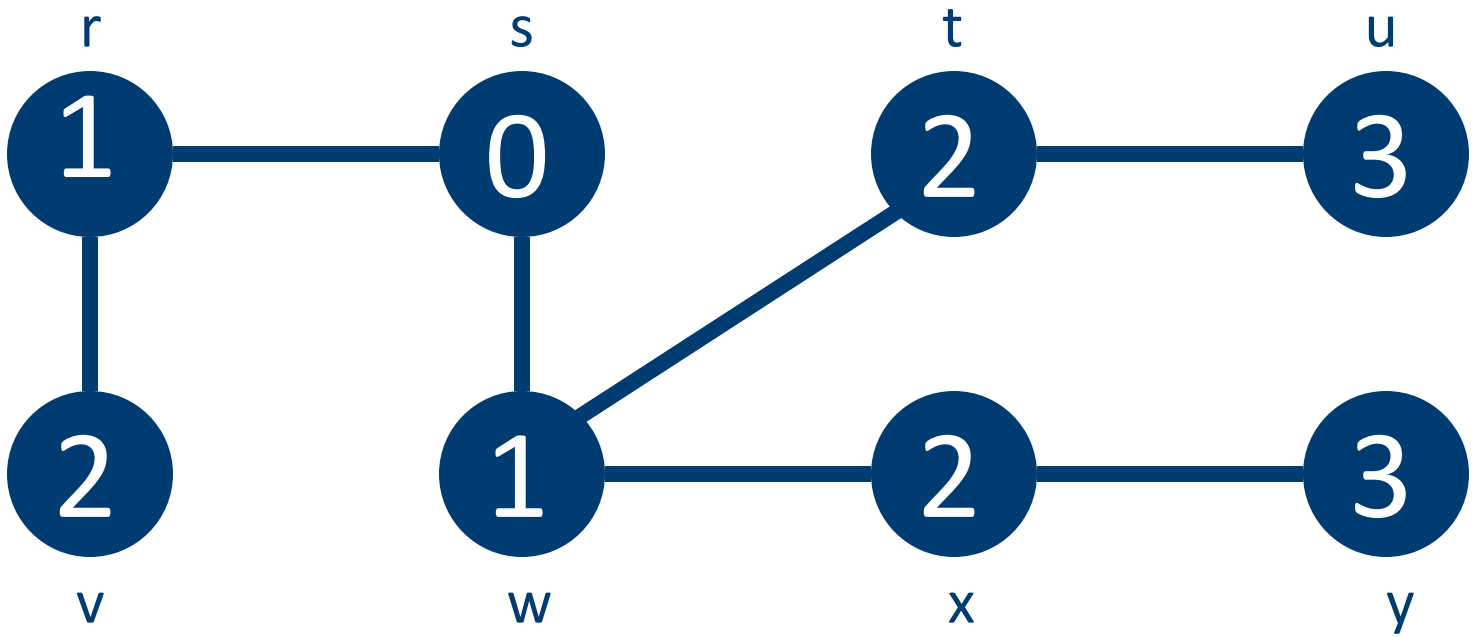
Q: y

BFS Example



Q: ∅

BFS Example



Set of edges $\{ (\pi.v, v): v \neq s\}$ forms a tree.

BFS Algorithm

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

$v.d$: distance from vertex v to the source s

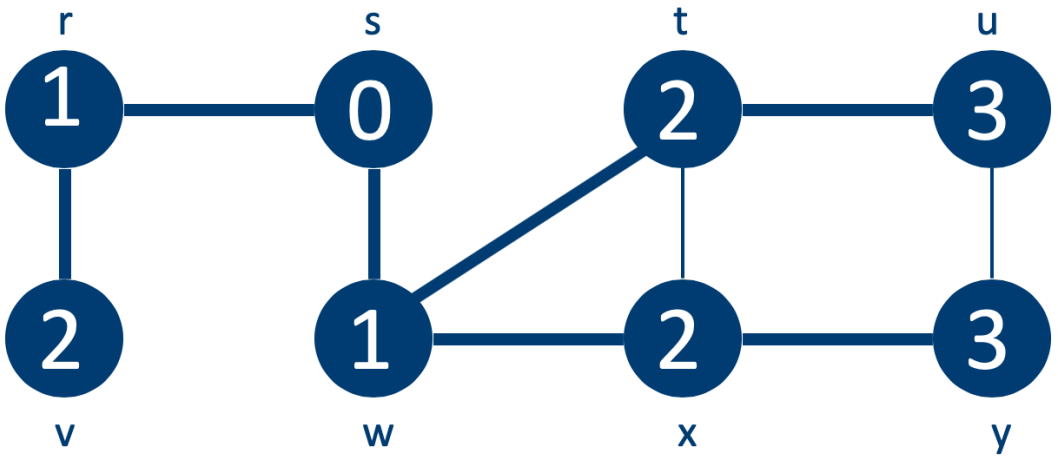
$v.\pi$: predecessor of vertex v along the shortest path from source s.

$Adj[u]$: list of neighbours of vertex u.

$$\Theta(V + E)$$

BFS Algorithm

```
PRINT-PATH(G, s, v)
1  if v == s
2      print s
3  elseif v.π == NIL
4      print "no path from" s "to" v "exists"
5  else PRINT-PATH(G, s, v.π)
6      print v
```

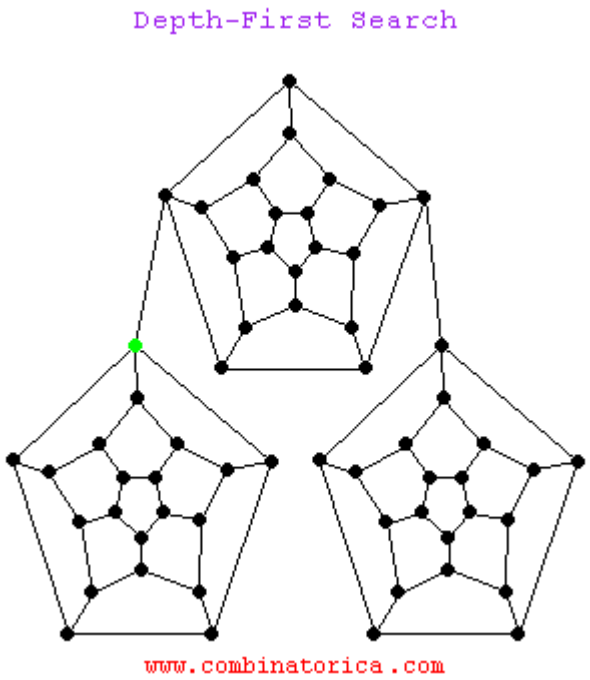


PRINT-PATH(G, s, y)

s w x y

Depth First Search (DFS)

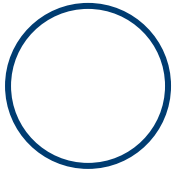
- **Input:**
  - Graph G=(V, E), either directed or undirected
- **Output:** Two *timestamps on each vertex:*
  - $v.d$ = discovery time
  - $v.f$ = finishing time
  - $v.\pi$ = predecessor of vertex v in the DFS.



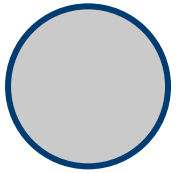
These will be useful for other algorithms later on !

Depth First Search (DFS)

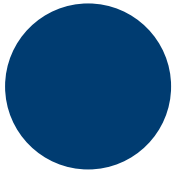
- To keep track of progress, depth first search colors each vertex white, gray, or black.
  - White vertices have not been discovered
    - All vertices start out white



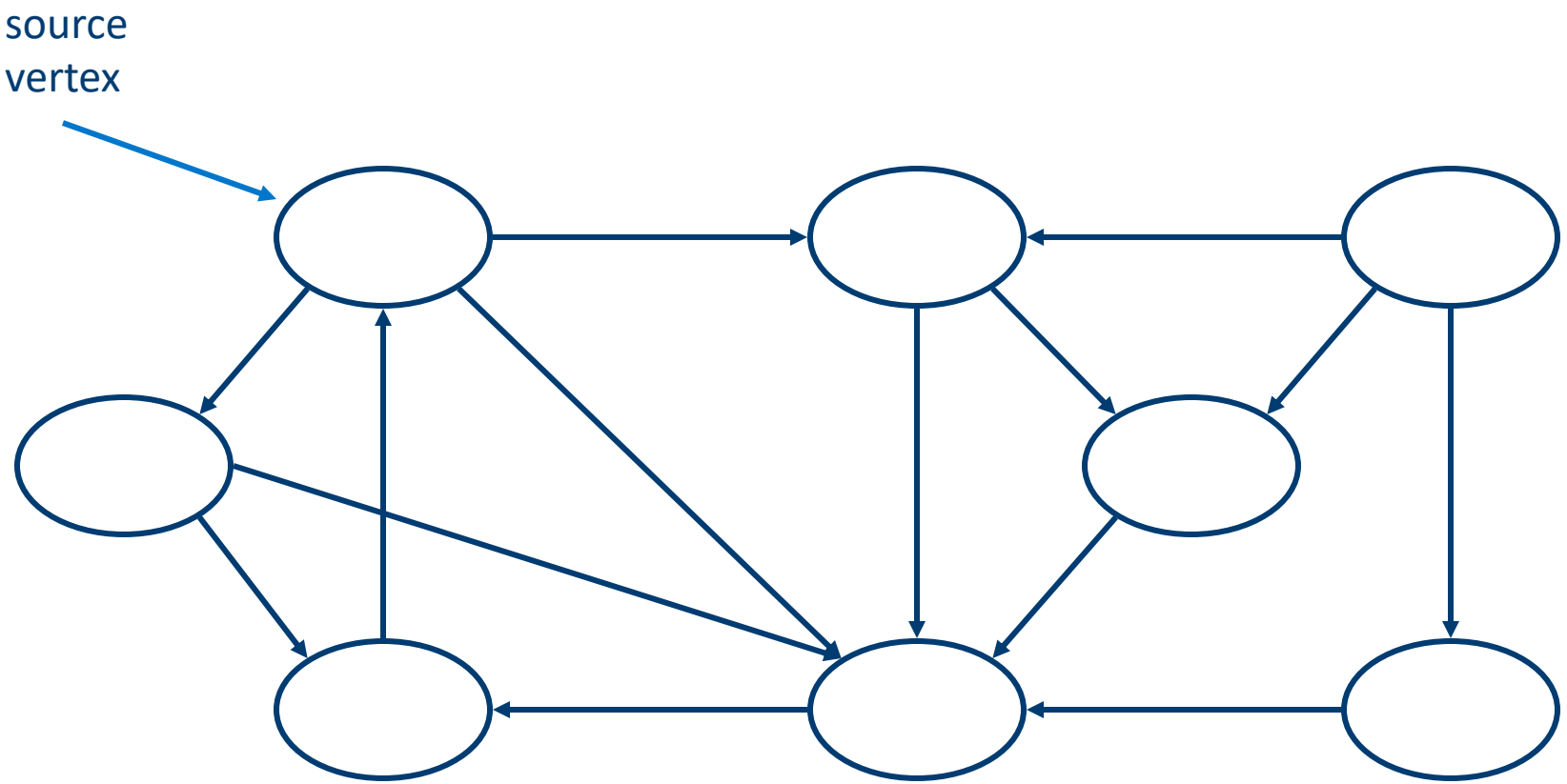
  - Grey vertices are discovered but not fully explored
    - They are visited but waiting for neighborhood visit



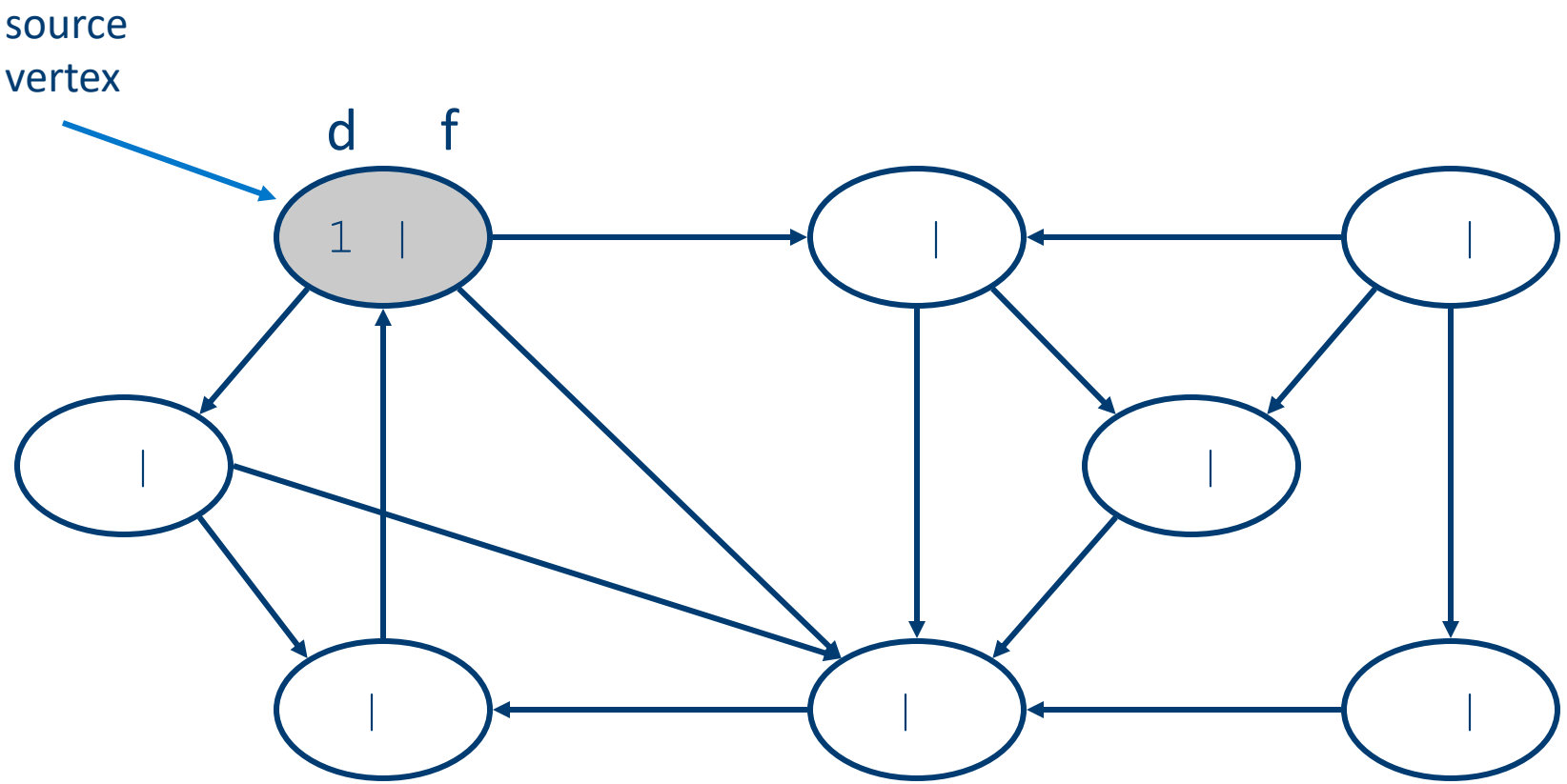
  - Black vertices are discovered and fully explored
    - Every edge leaving the vertex has been explored



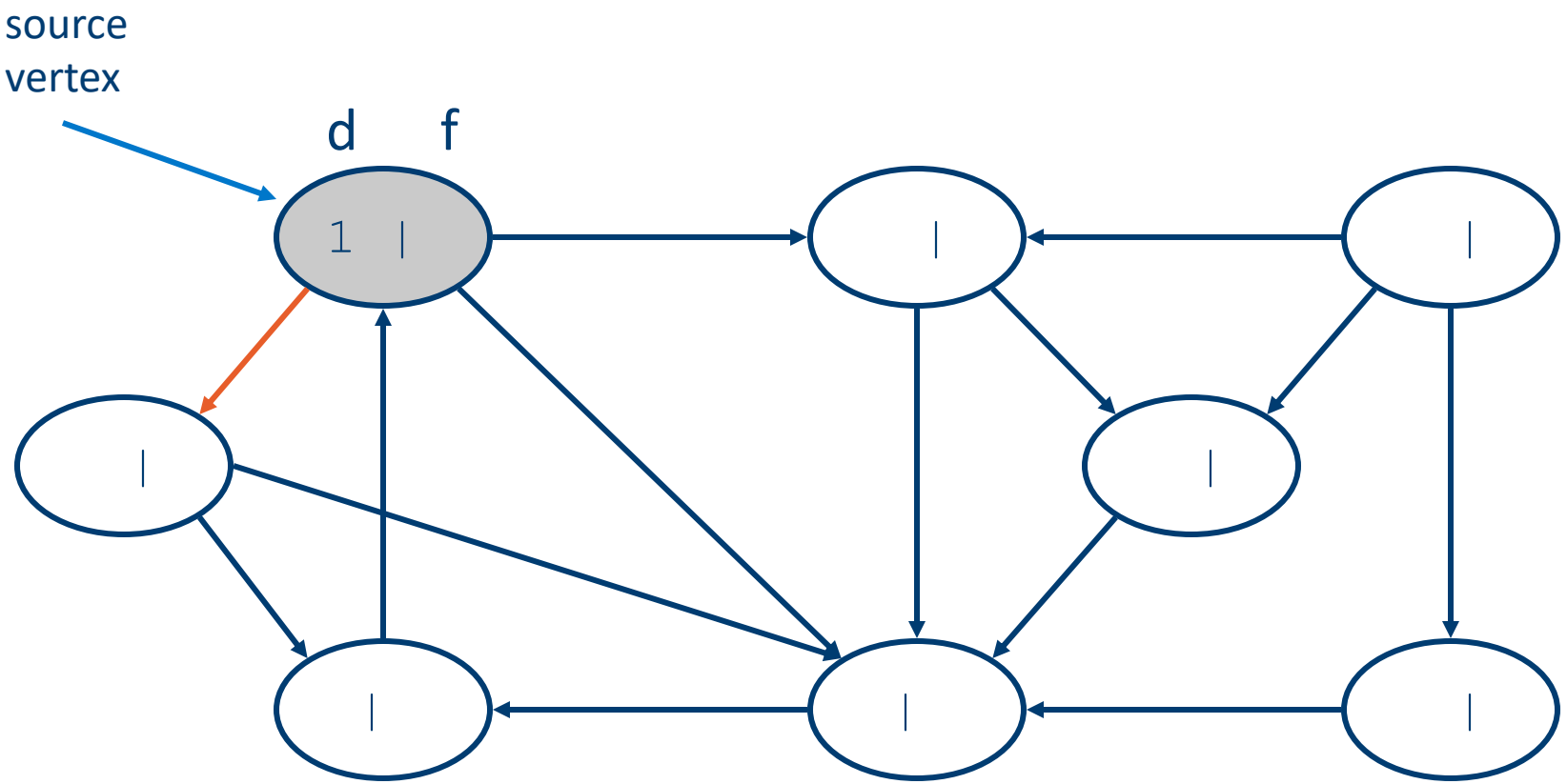
DFS Example



DFS Example

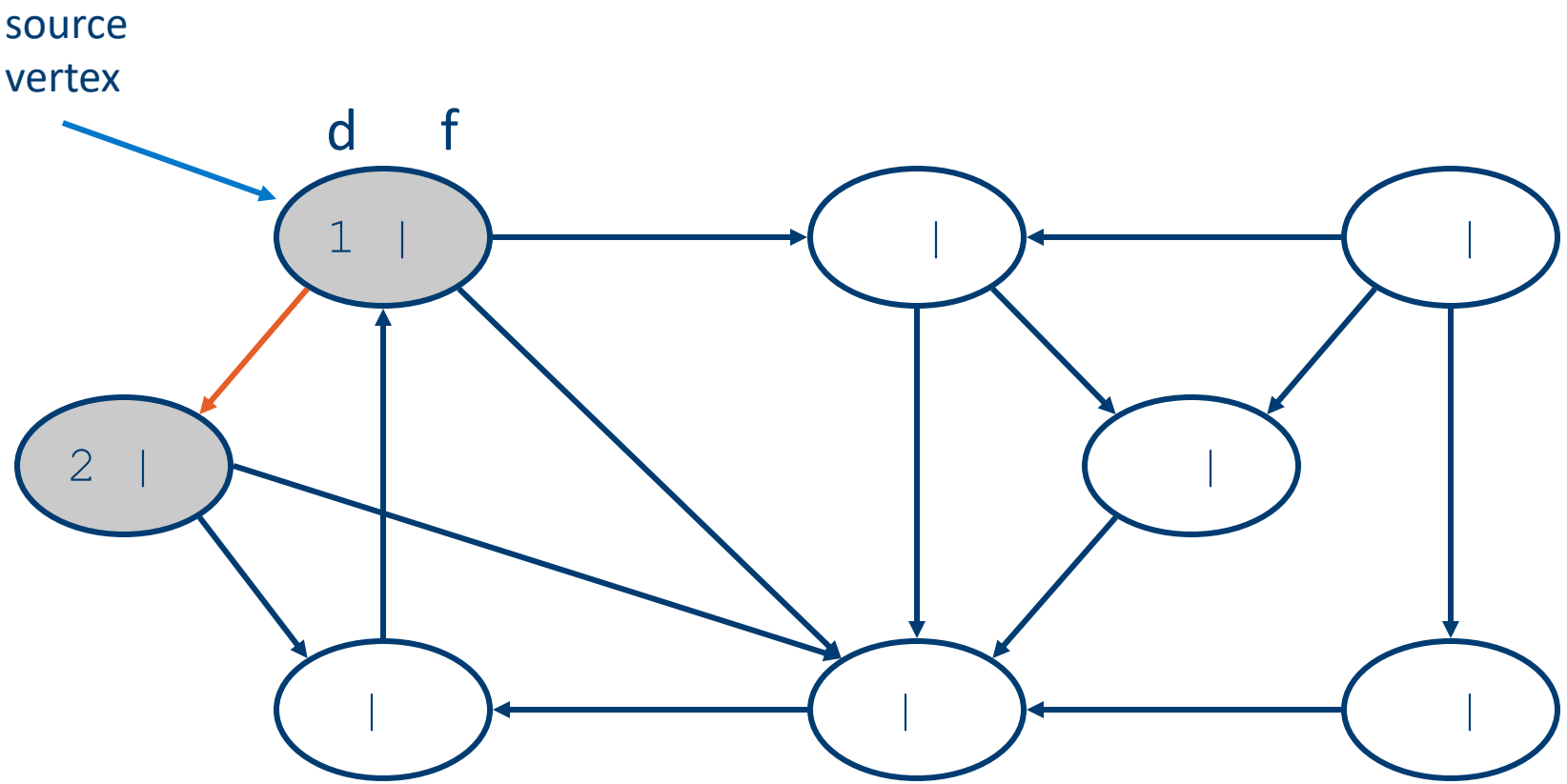


DFS Example

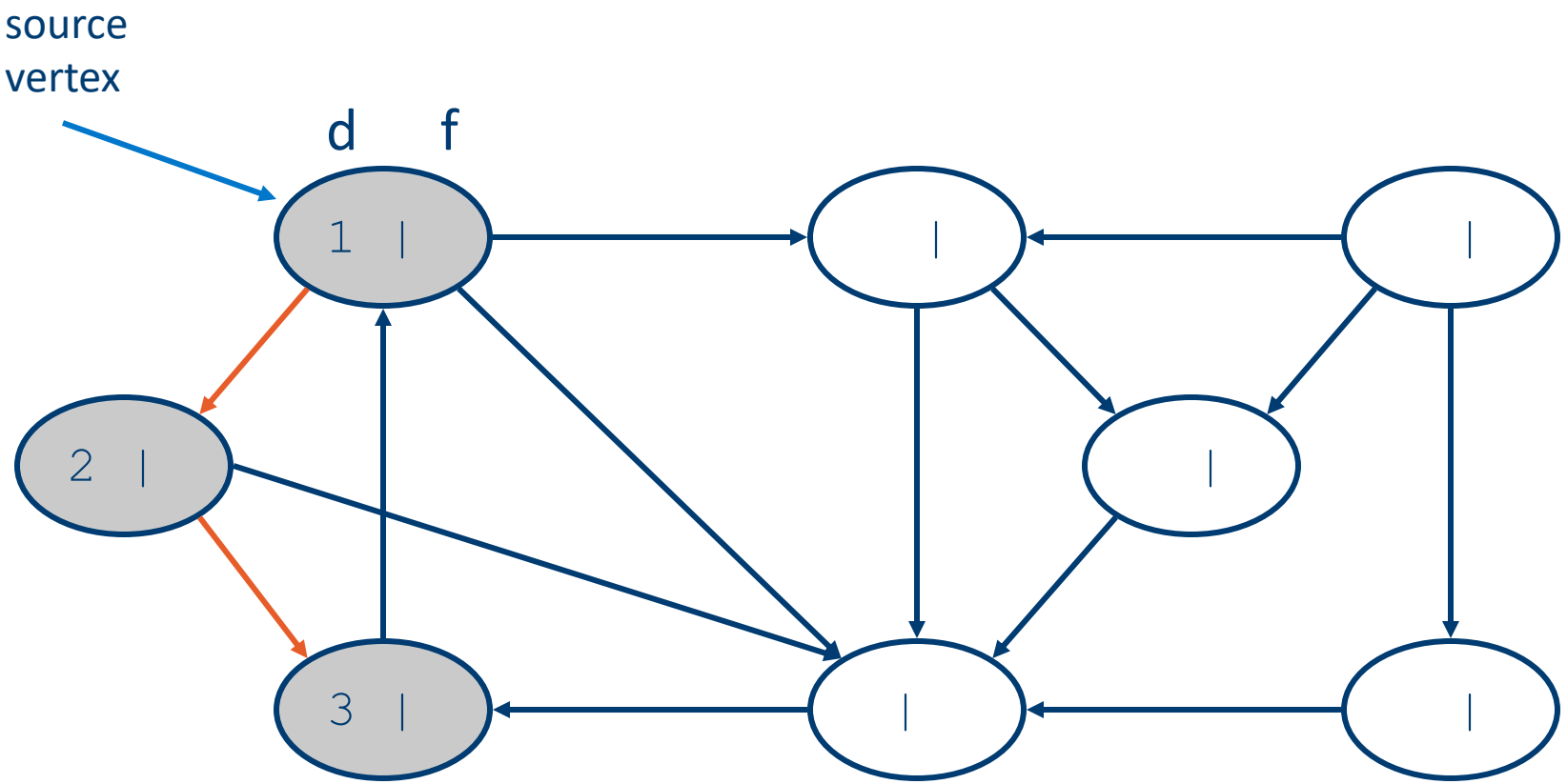


Tree Edge: visiting new vertex (gray → white)

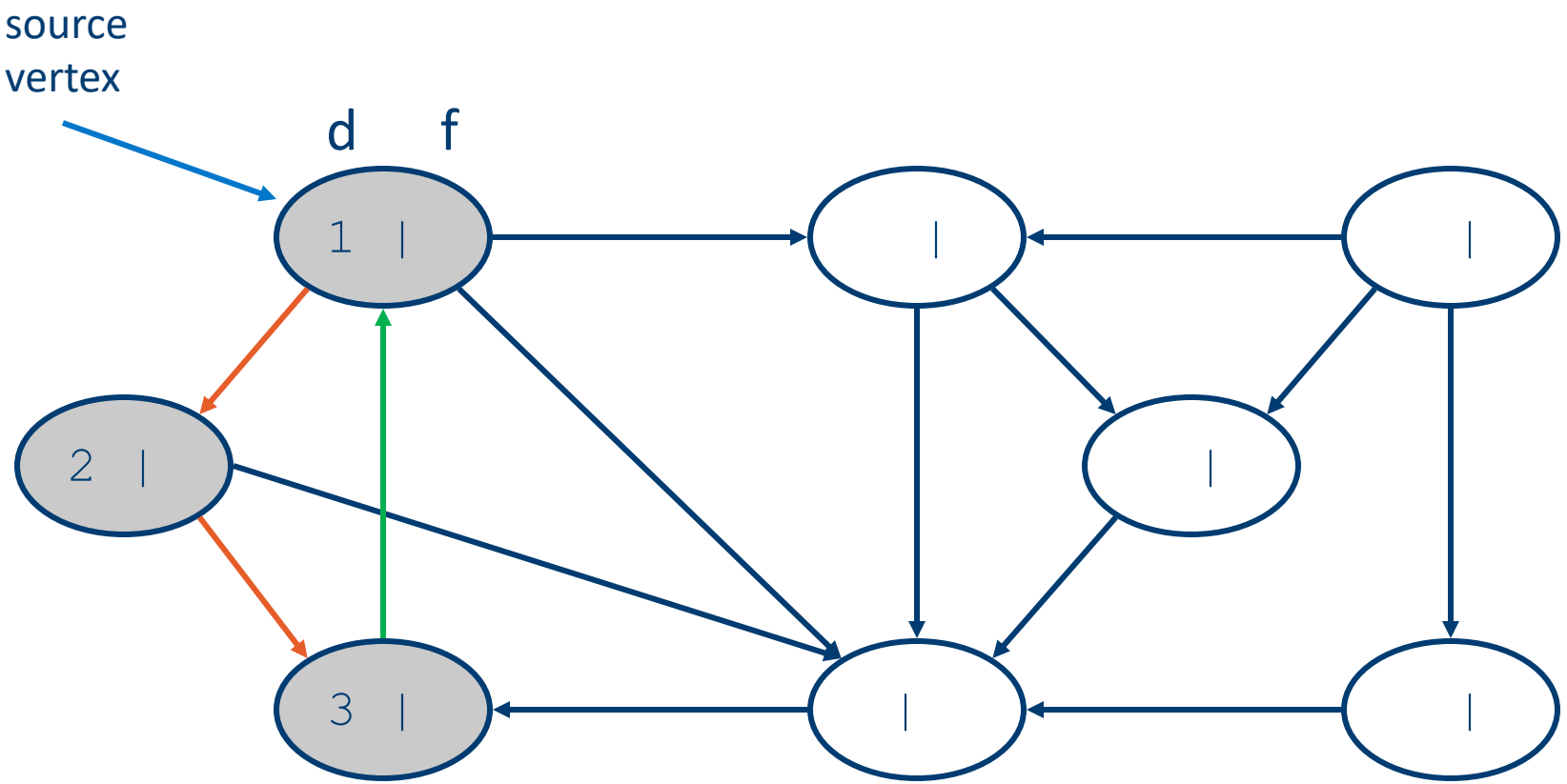
DFS Example



DFS Example

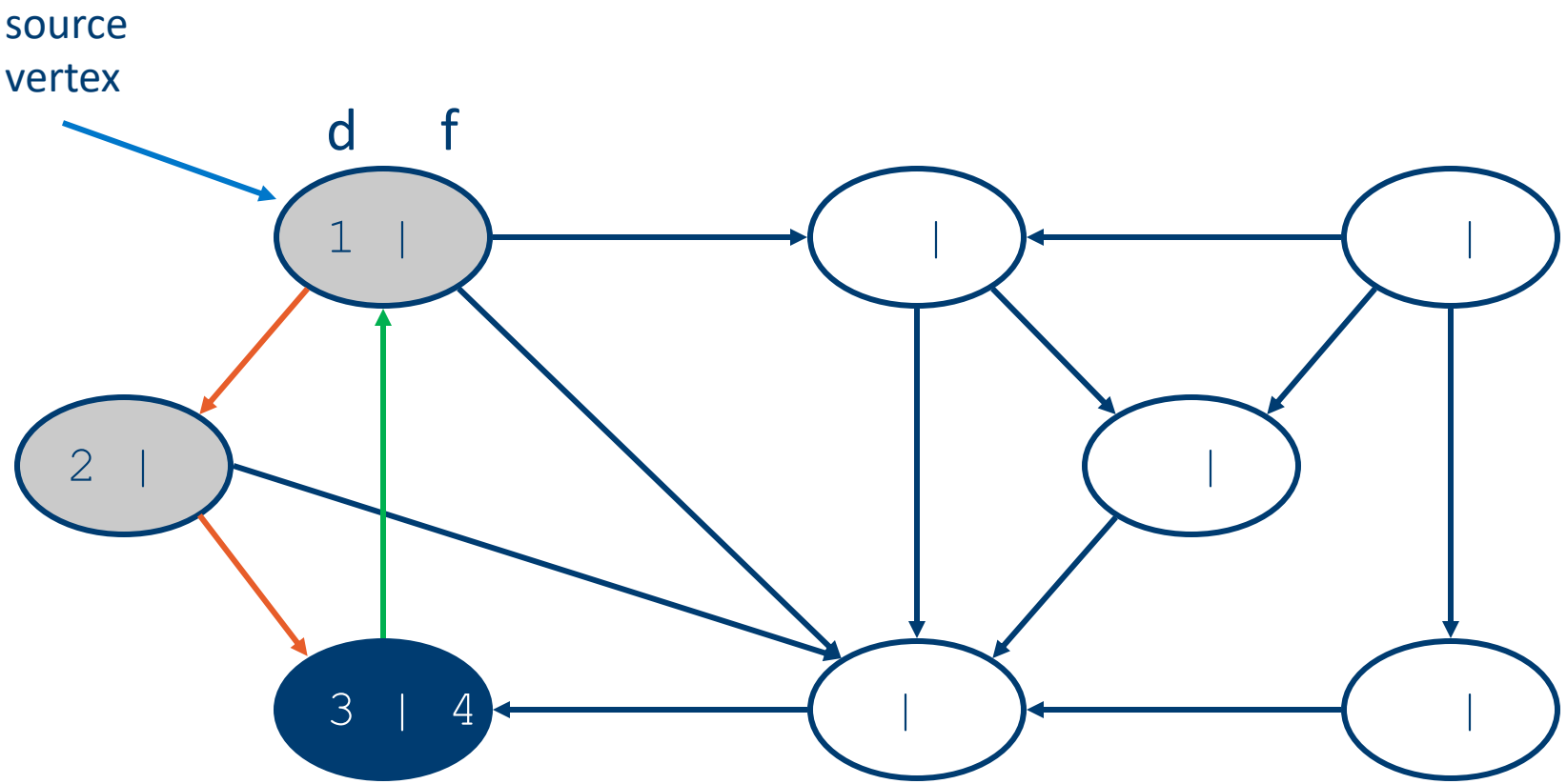


DFS Example

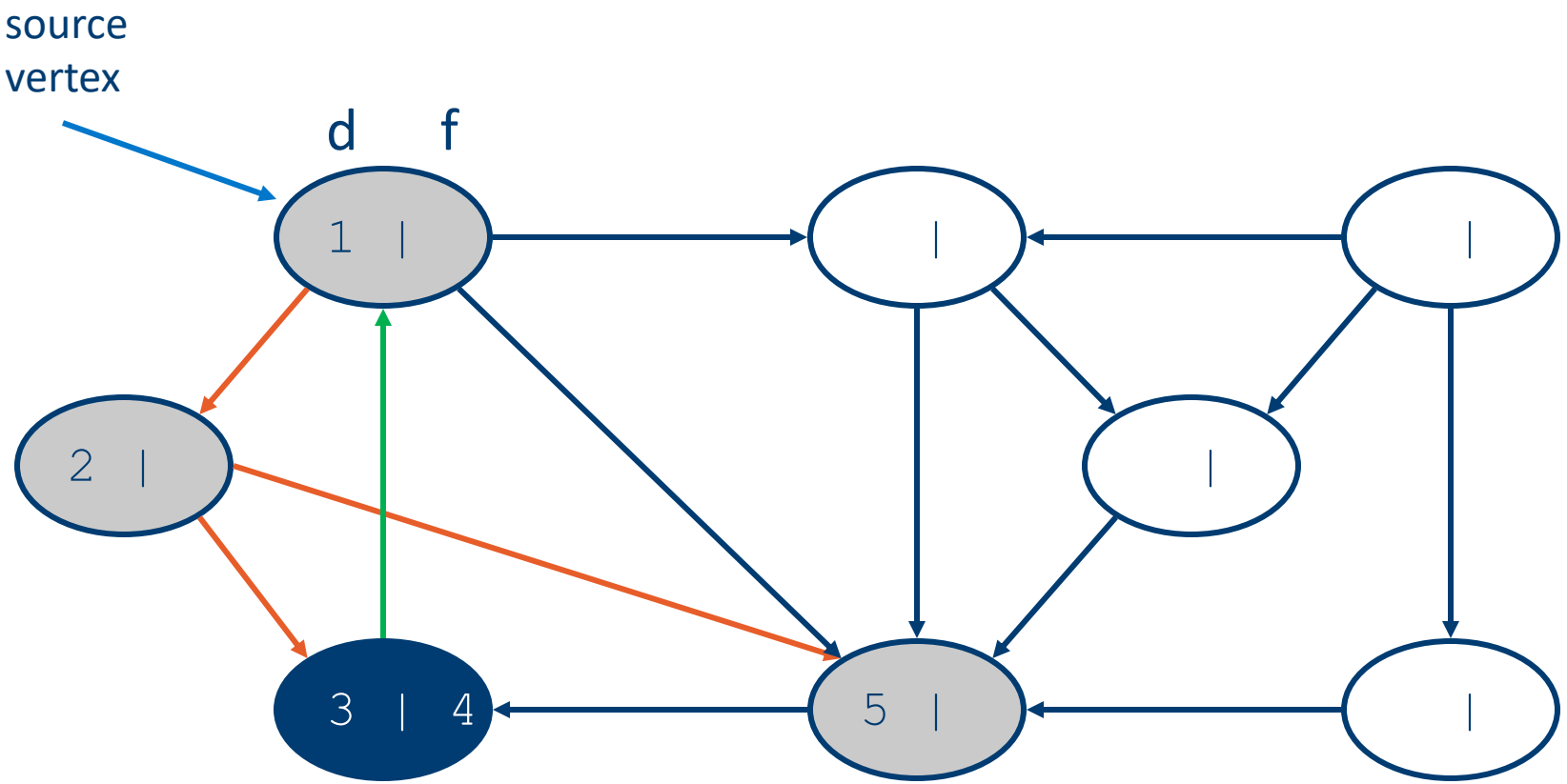


Back Edge: from descendent to ancestor (gray → gray)

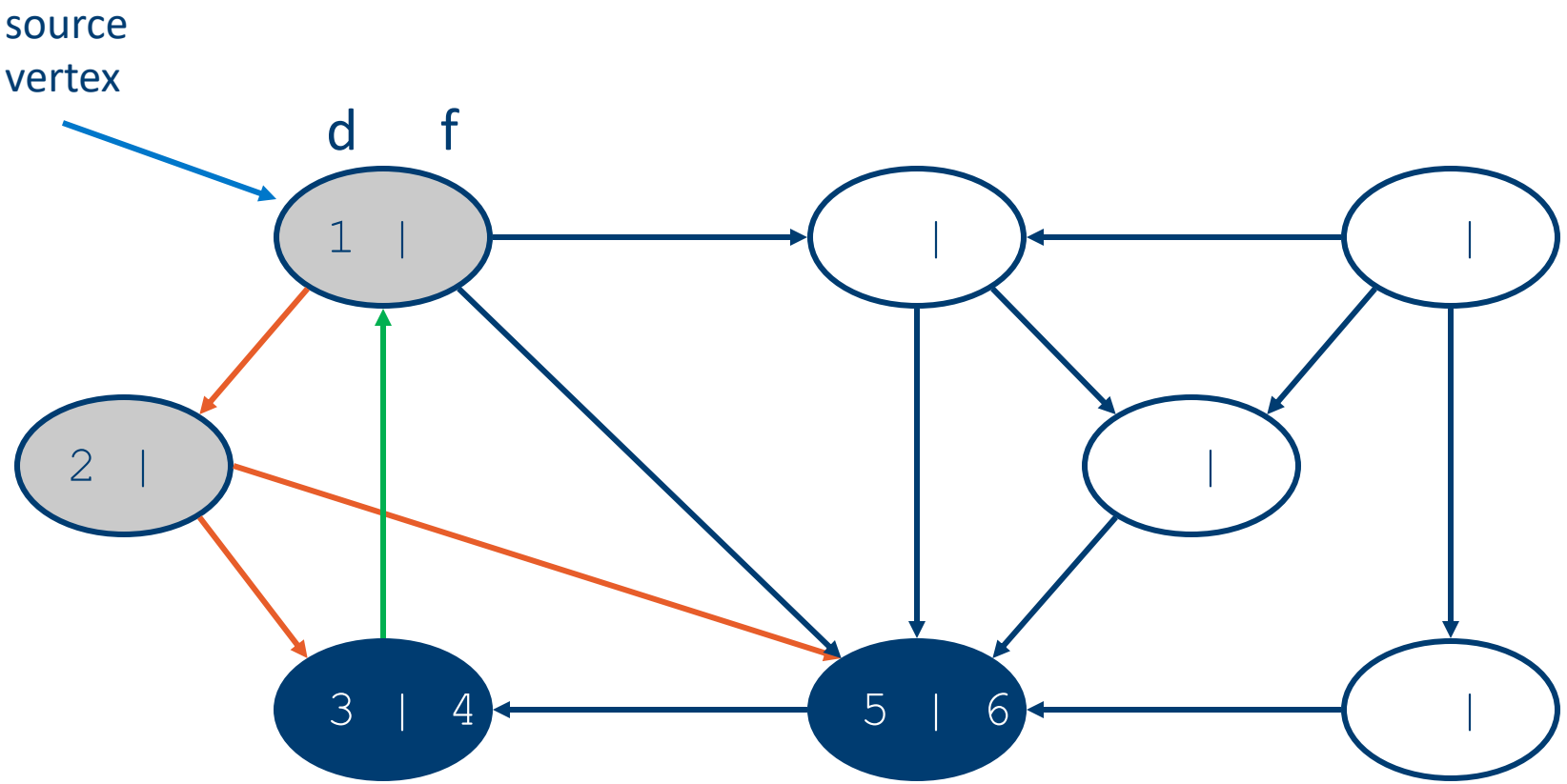
DFS Example



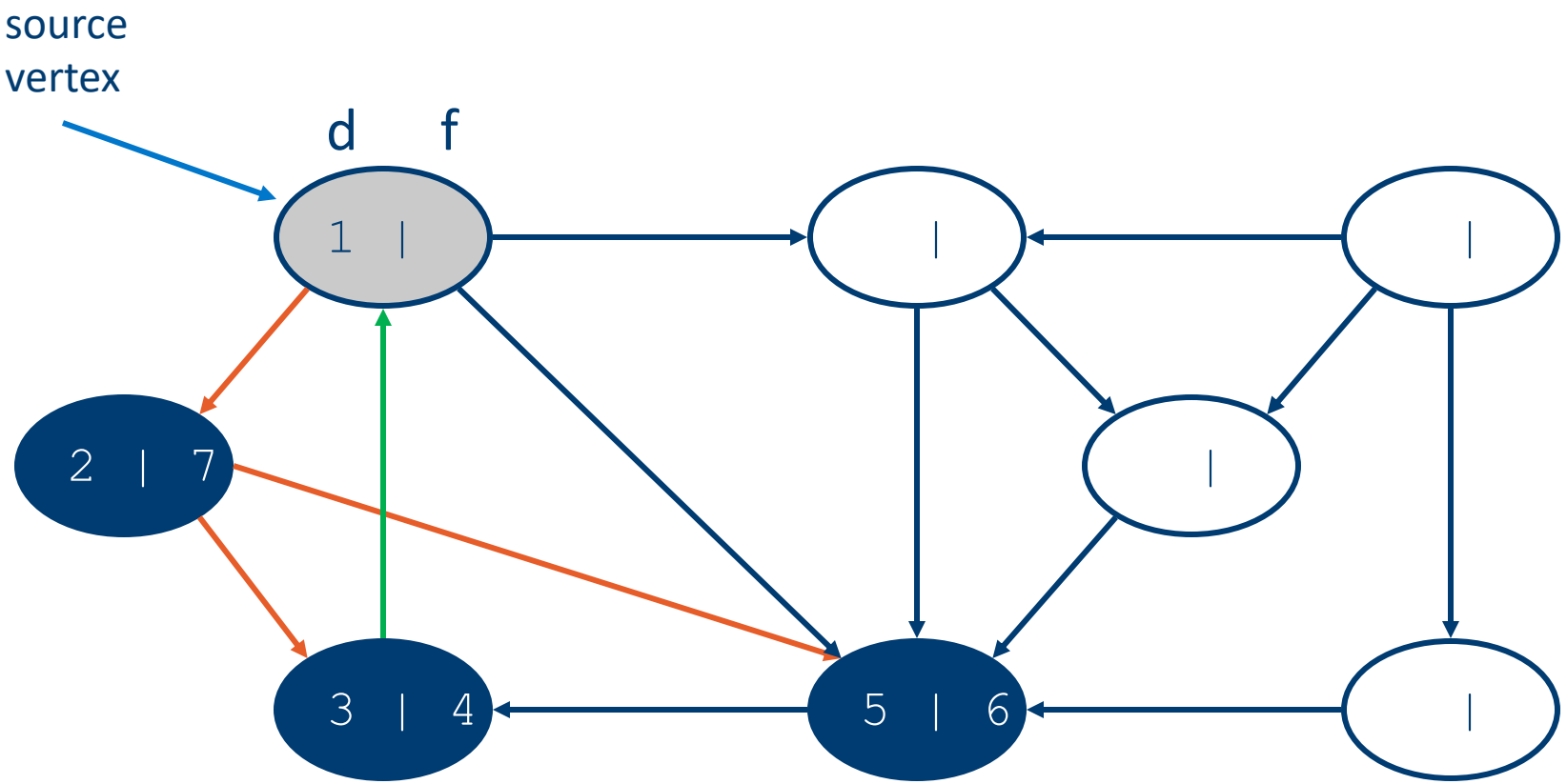
DFS Example



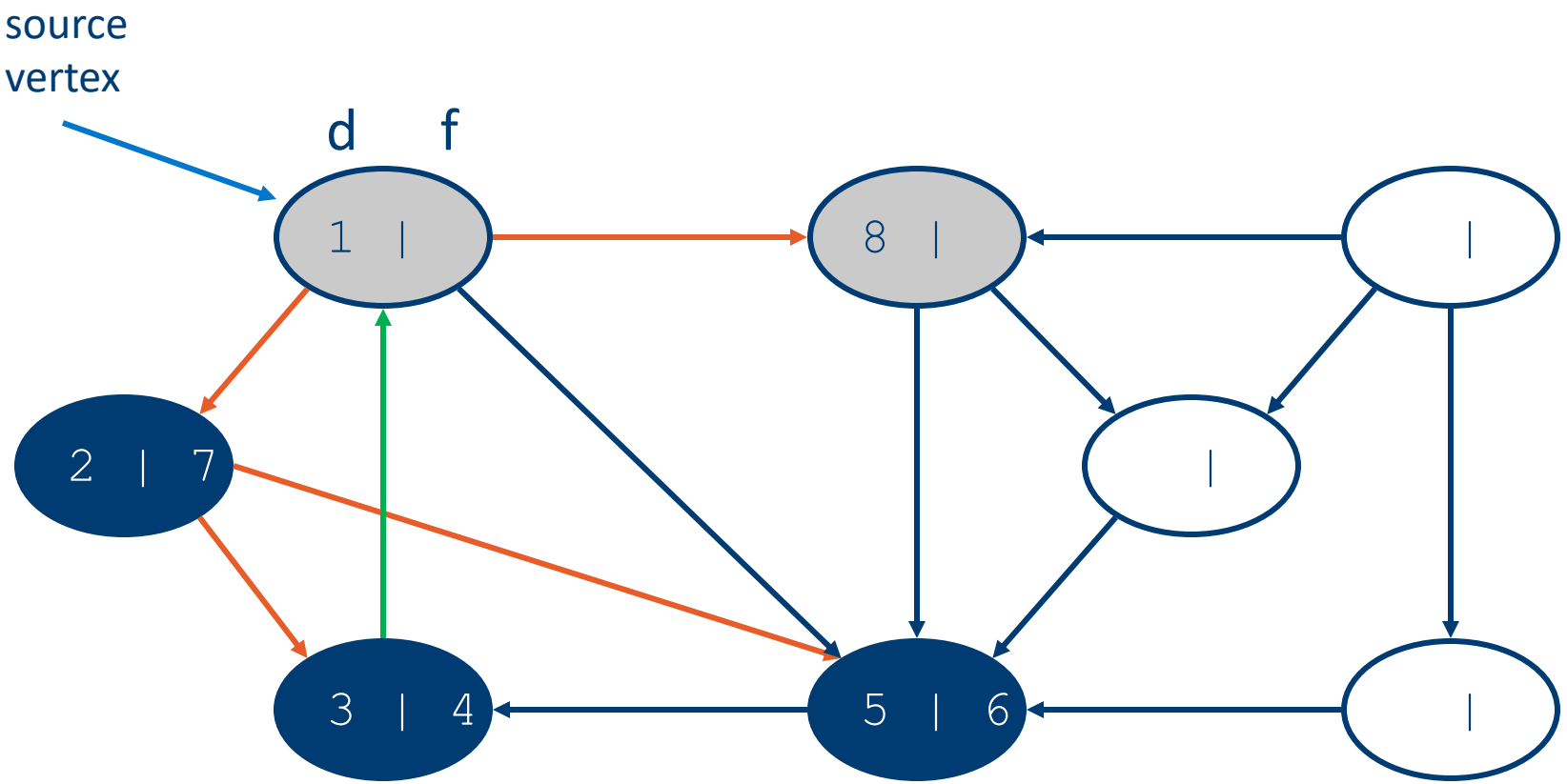
DFS Example



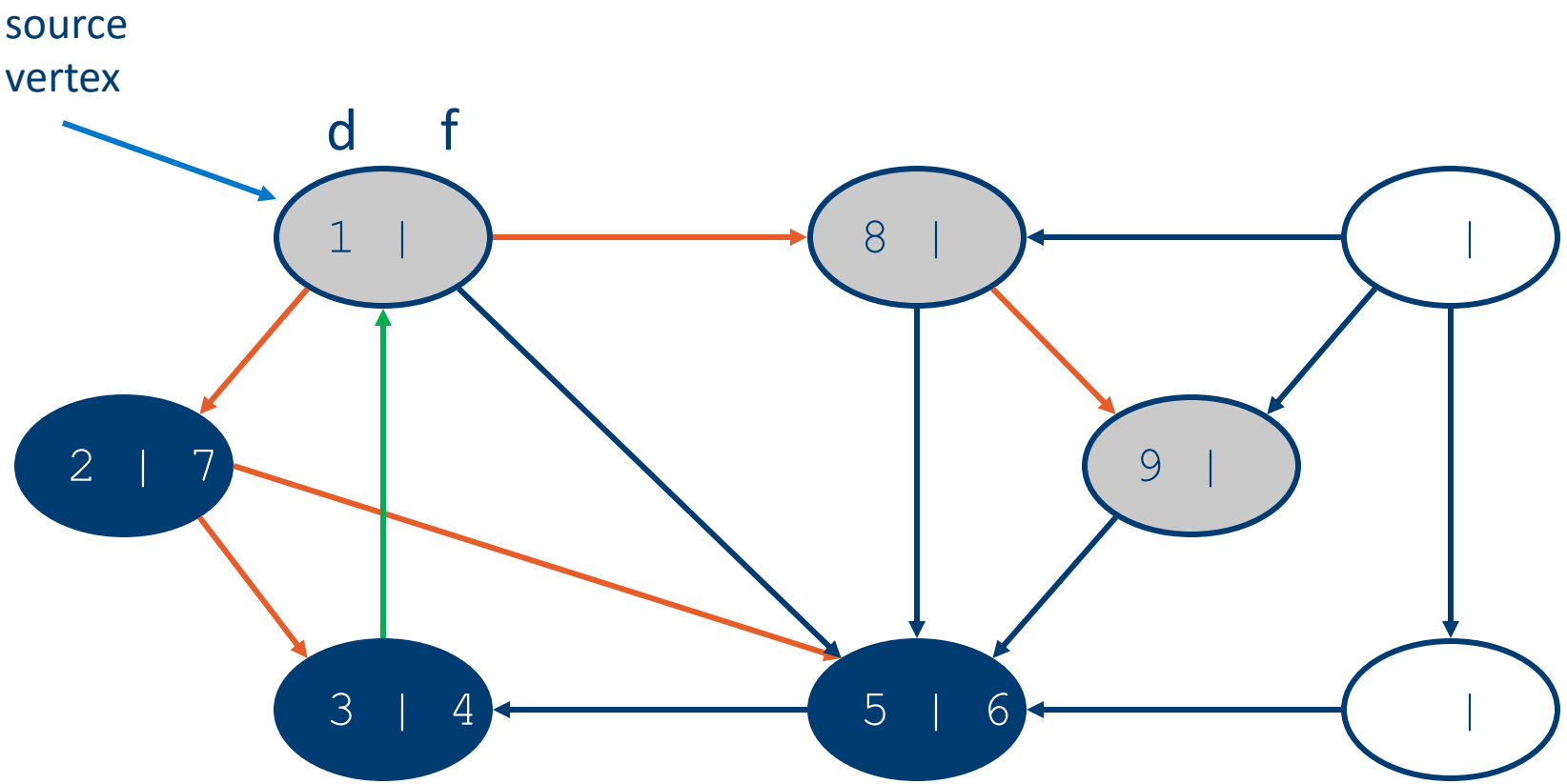
DFS Example



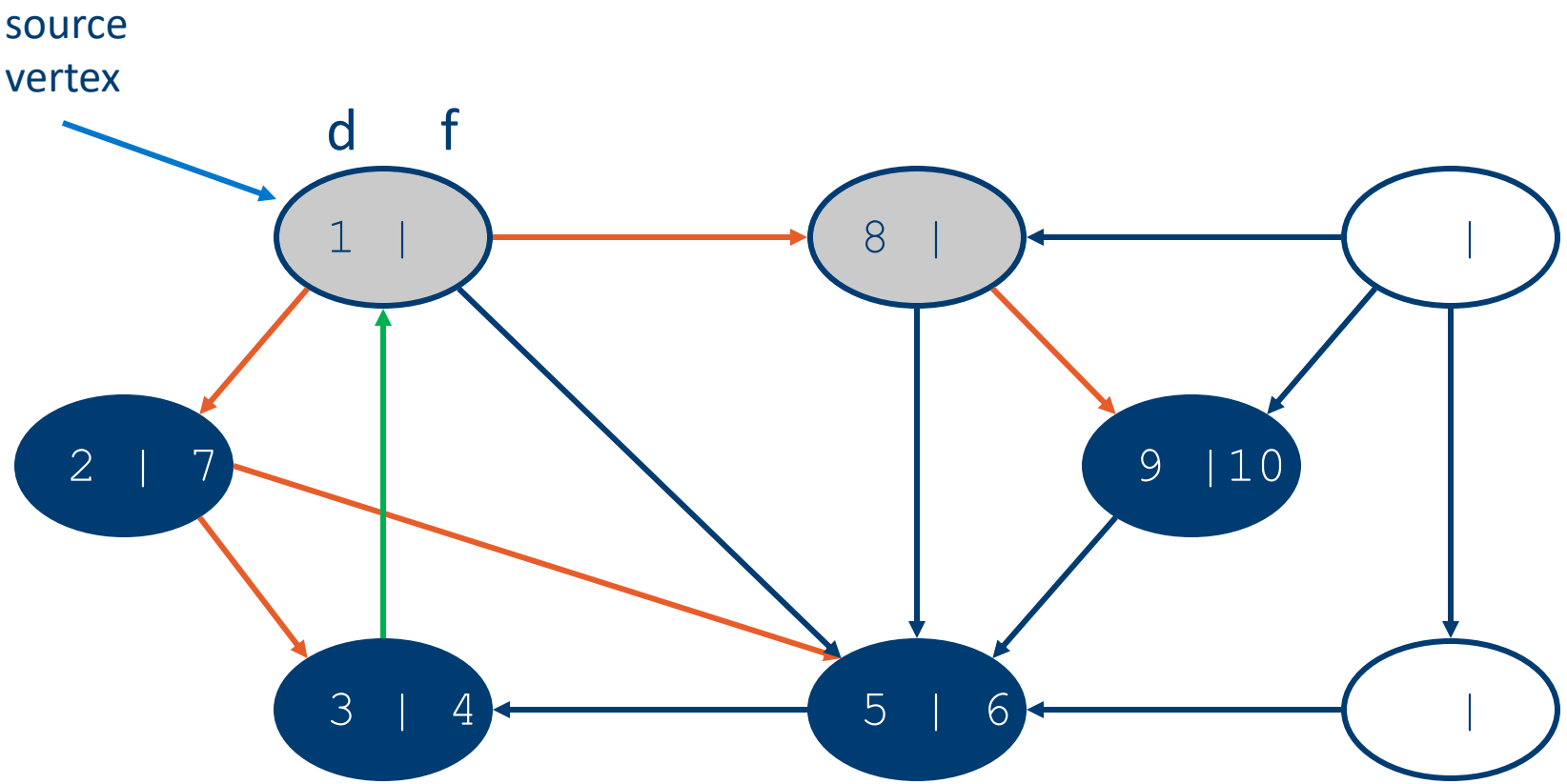
DFS Example



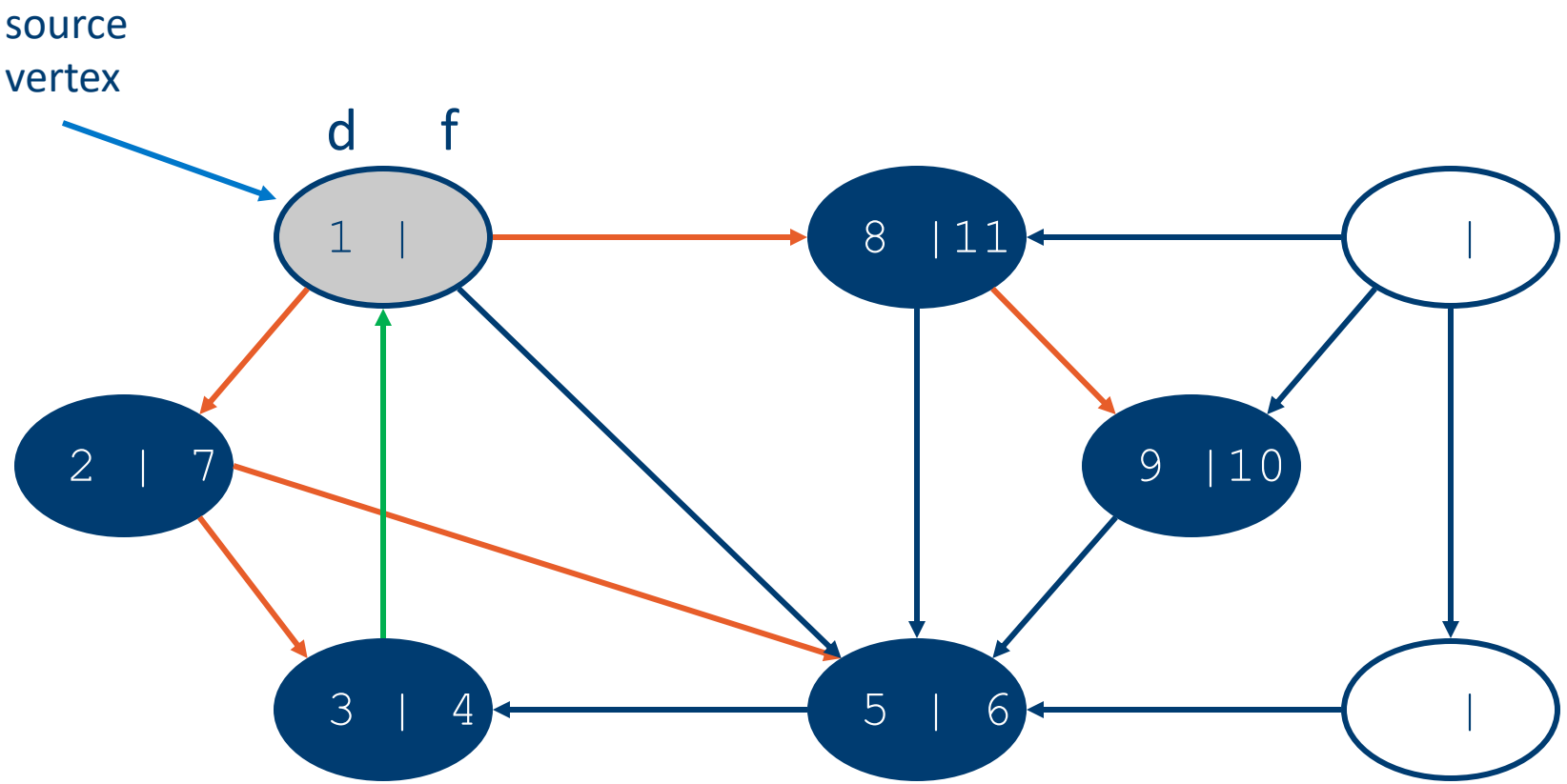
DFS Example



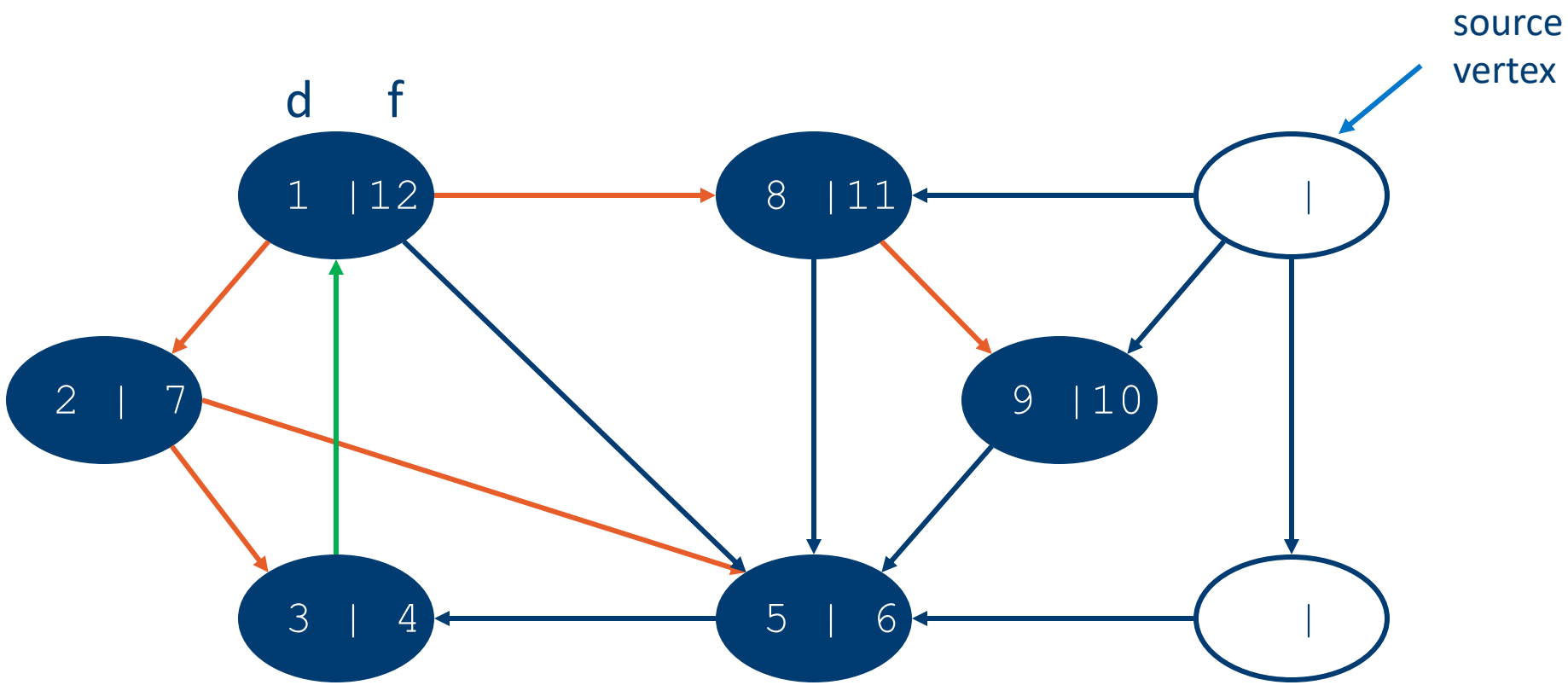
DFS Example



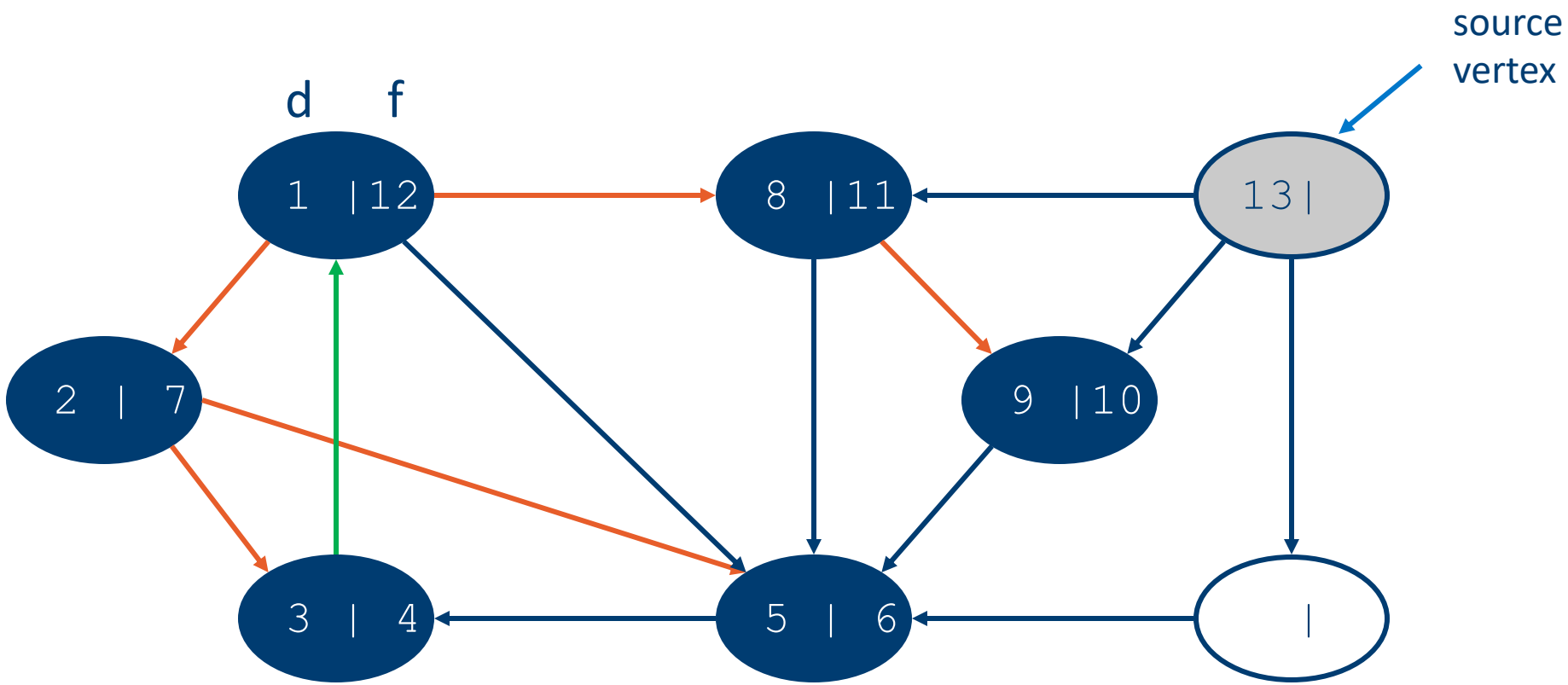
DFS Example



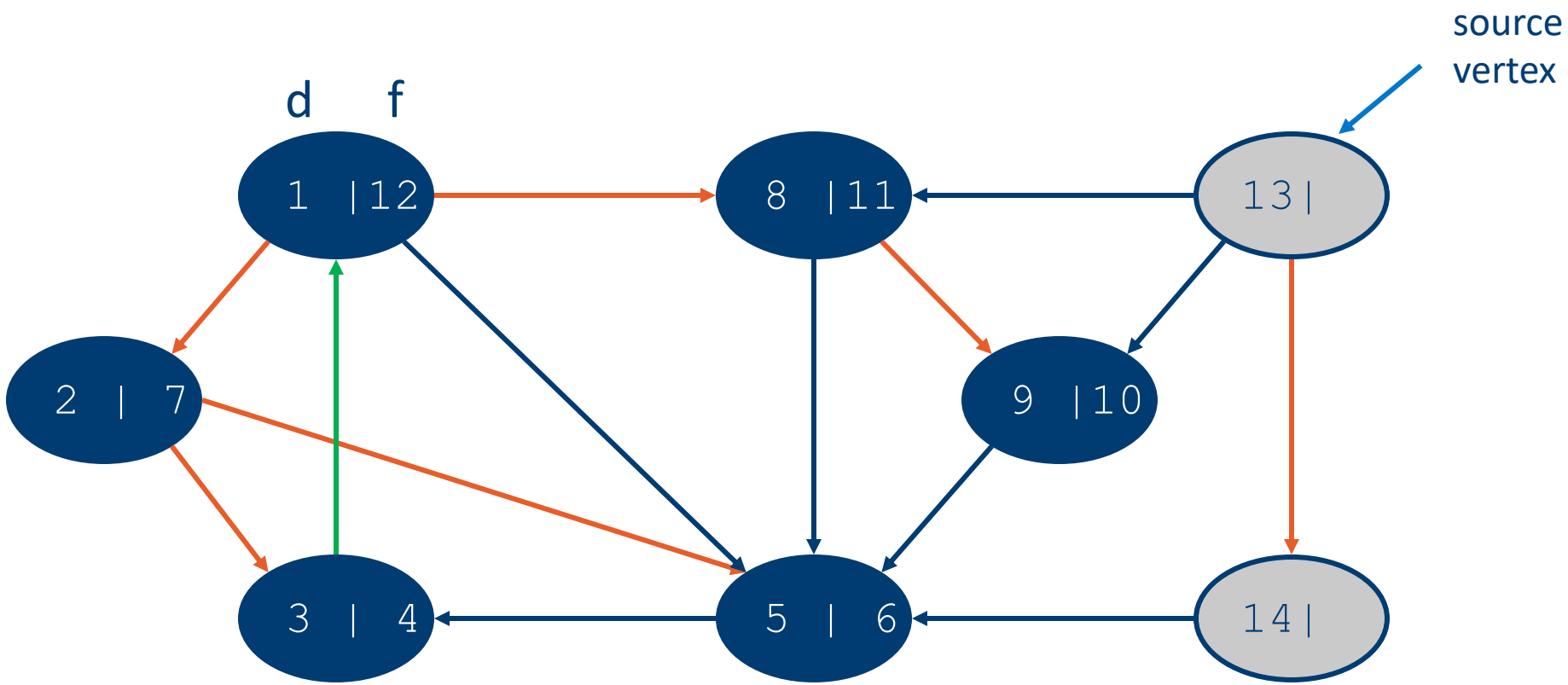
DFS Example



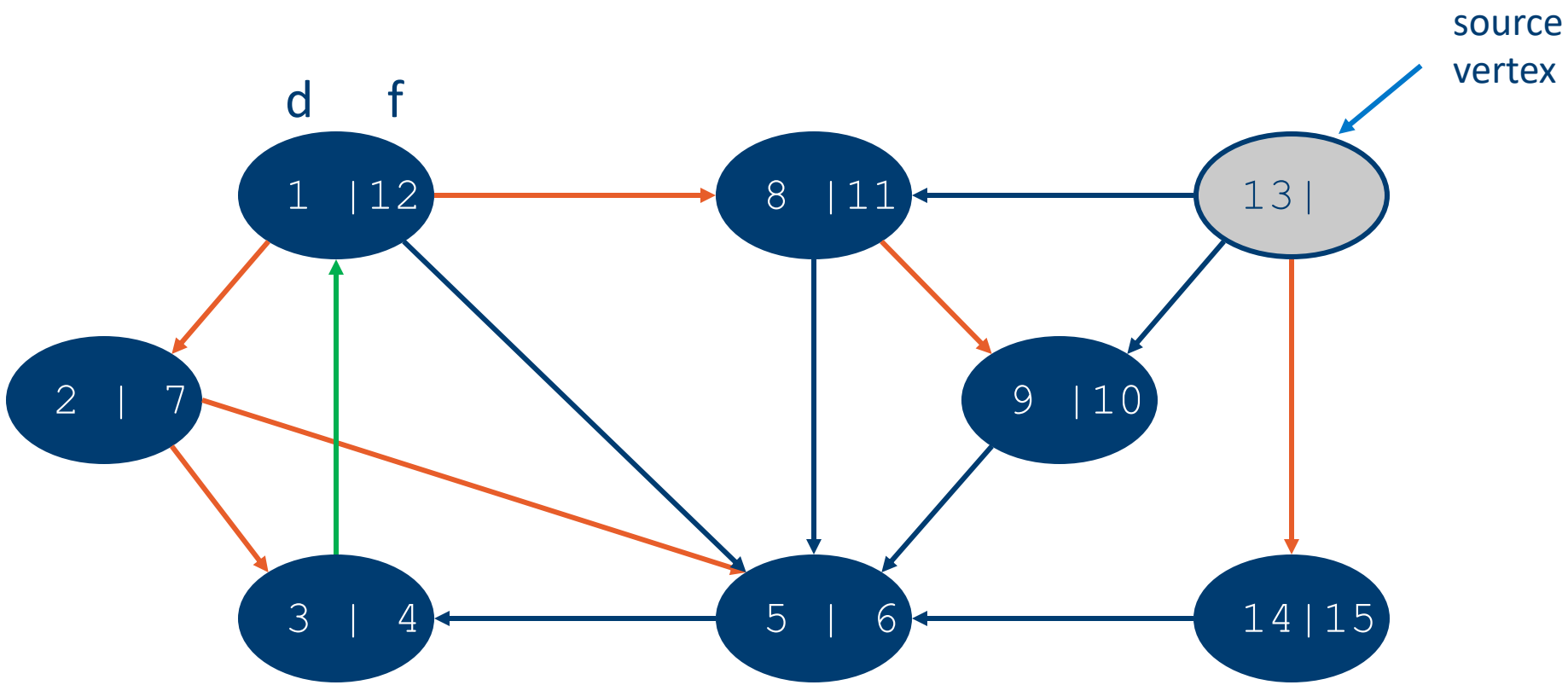
DFS Example



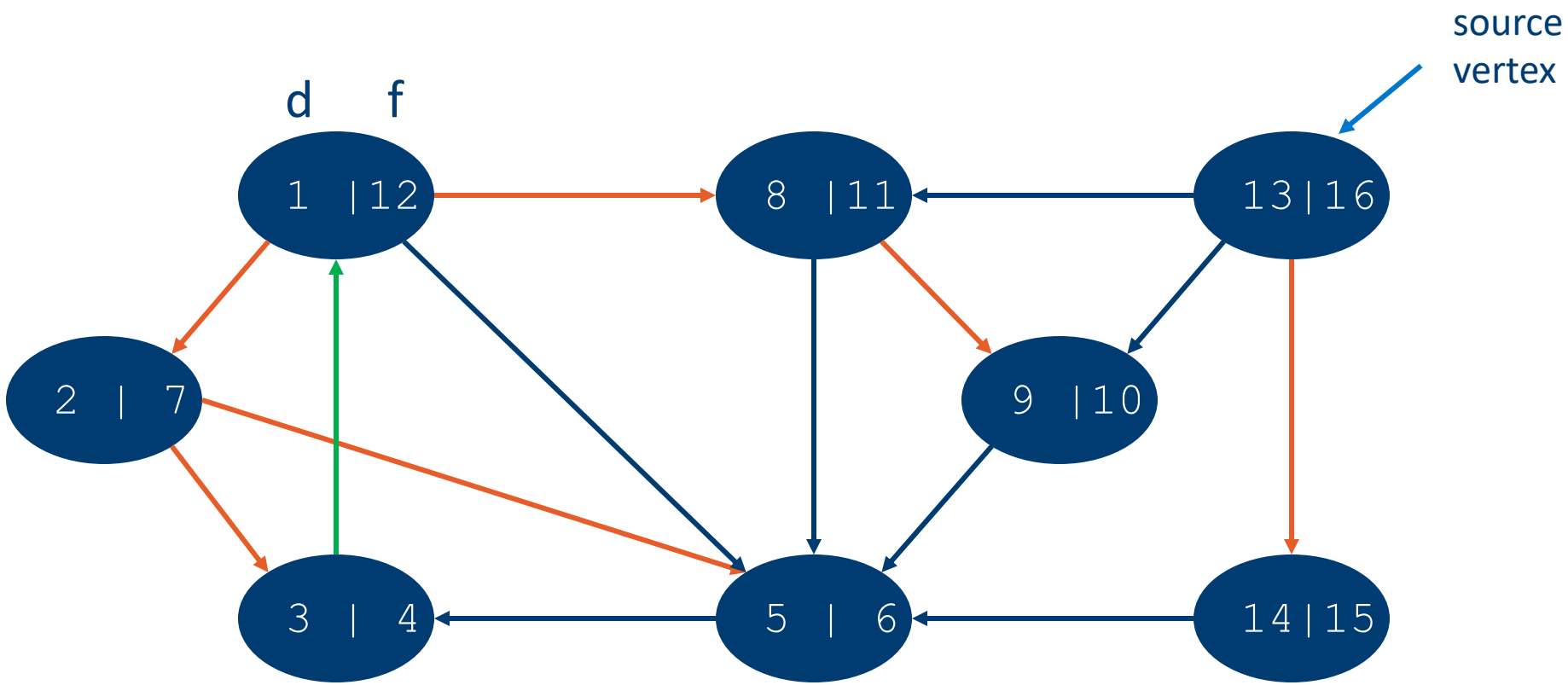
DFS Example



DFS Example



DFS Example



DFS Algorithm

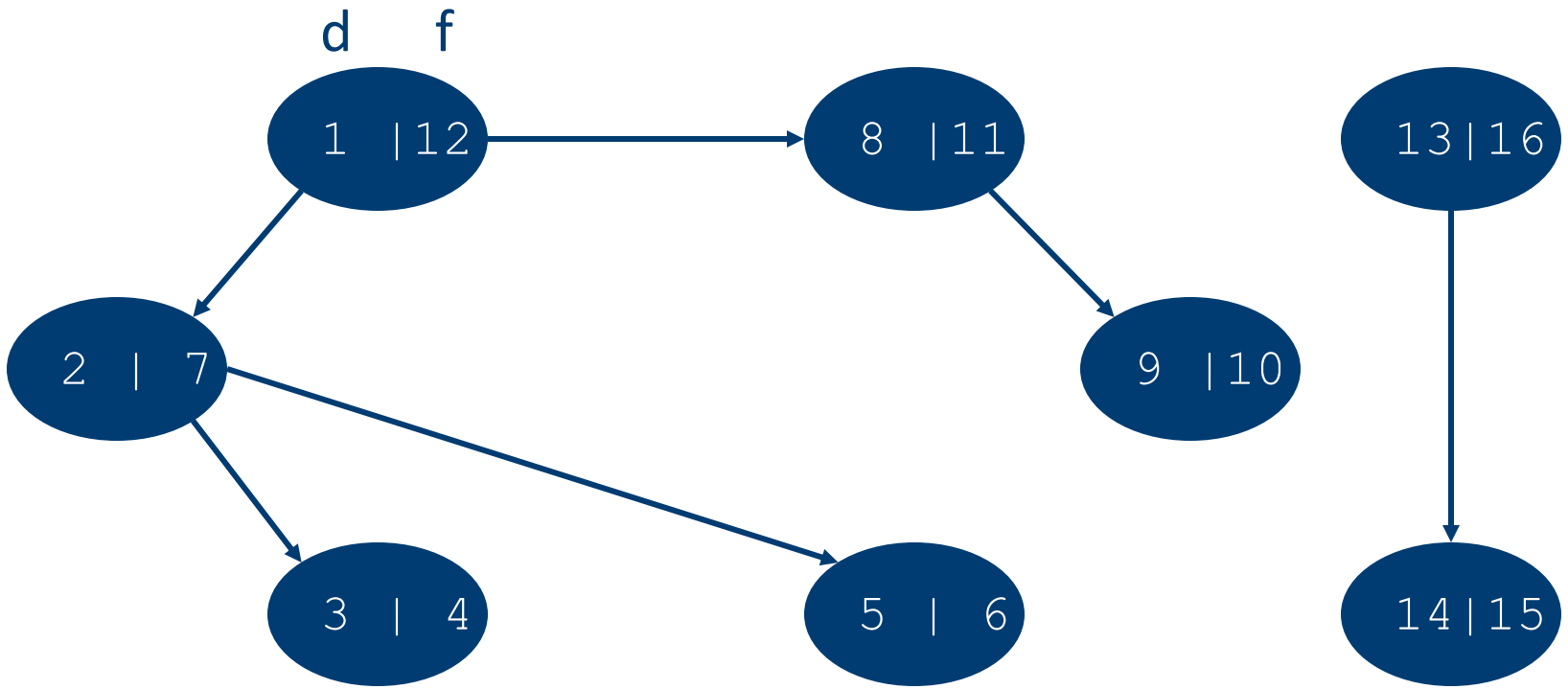
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

$$\Theta(V + E)$$

DFS Algorithm

```
DFS-VISIT(G, u)
 1  time = time + 1                  // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]            // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK                  // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

DFS Example



Wrap-up

- We learned:
  - Fundamental some concepts in graph theory
  - Search in graph
    - BFS
    - DFS