Design and analysis of algorithm

prepared By:
13cse035:Hemanshi maheta
13cse038:Satya parsana
13cse047:Jignesh rathod

Topics

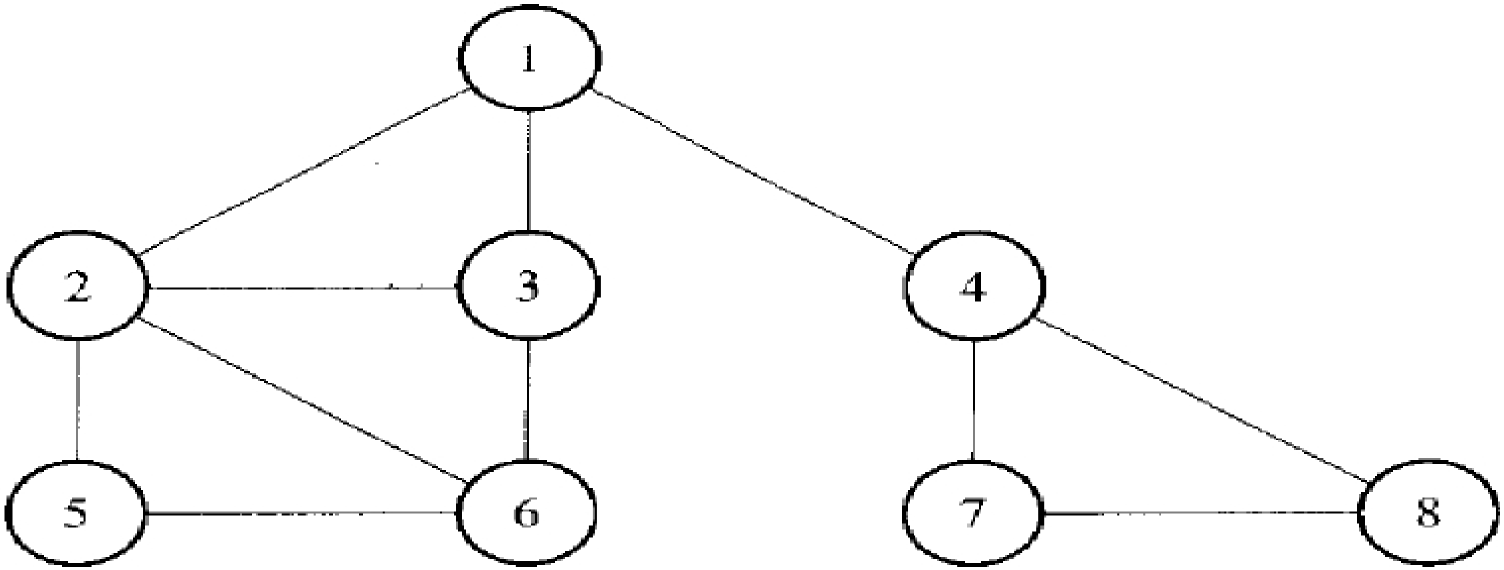
- Depth First Search (DFS)
- Breadth First Search (BFS)

DEPTH-FIRST SEARCH: UNDIRECTED GRAPHS

- Let **G = (N , A)** be an undirected graph all of whose nodes we wish to visit.
- Suppose it is somehow possible to mark a node to show it has already been visited.
- To carry out a depth-first traversal of the graph, choose any node **v** ϵ **N** as the starting point.
- Mark this node to show it has been visited. Next, if there is a node adjacent to **v** that has not yet been visited, choose this node as a new starting point and call the depth-first search procedure recursively.

- On return from the recursive call, if there is another node adjacent to **v** that has not been visited, choose this node as the next starting point, call the procedure recursively once again, and so on.
- When all the nodes adjacent to **v** are marked, the search starting at **v** is finished.
- If there remain any nodes of G that have not been visited,choose any one of them as a new starting point, and call the procedure yet again.
- Continue thus until all the nodes of G are marked. Here is the recursive algorithm

EXAMPLE:



1.	*dfs*(1)	initial call
2.	*dfs*(2)	recursive call
3.	*dfs*(3)	recursive call
4.	*dfs*(6)	recursive call
5.	*dfs*(5)	recursive call; progress is blocked
6.	*dfs*(4)	a neighbour of node 1 has not been visited
7.	*dfs*(7)	recursive call
8.	*dfs*(8)	recursive call; progress is blocked
9.	there are no more nodes to visit	

procedure $dfsearch(G)$

    for each $v \in N$ **do** $mark[v] \leftarrow$ *not-visited*

    for each $v \in N$ **do**

        if $mark[v] \neq visited$ **then** $dfs(v)$

procedure $dfs(v)$

    {Node v has not previously been visited}

    $mark[v] \leftarrow visited$

    for each node w adjacent to v **do**

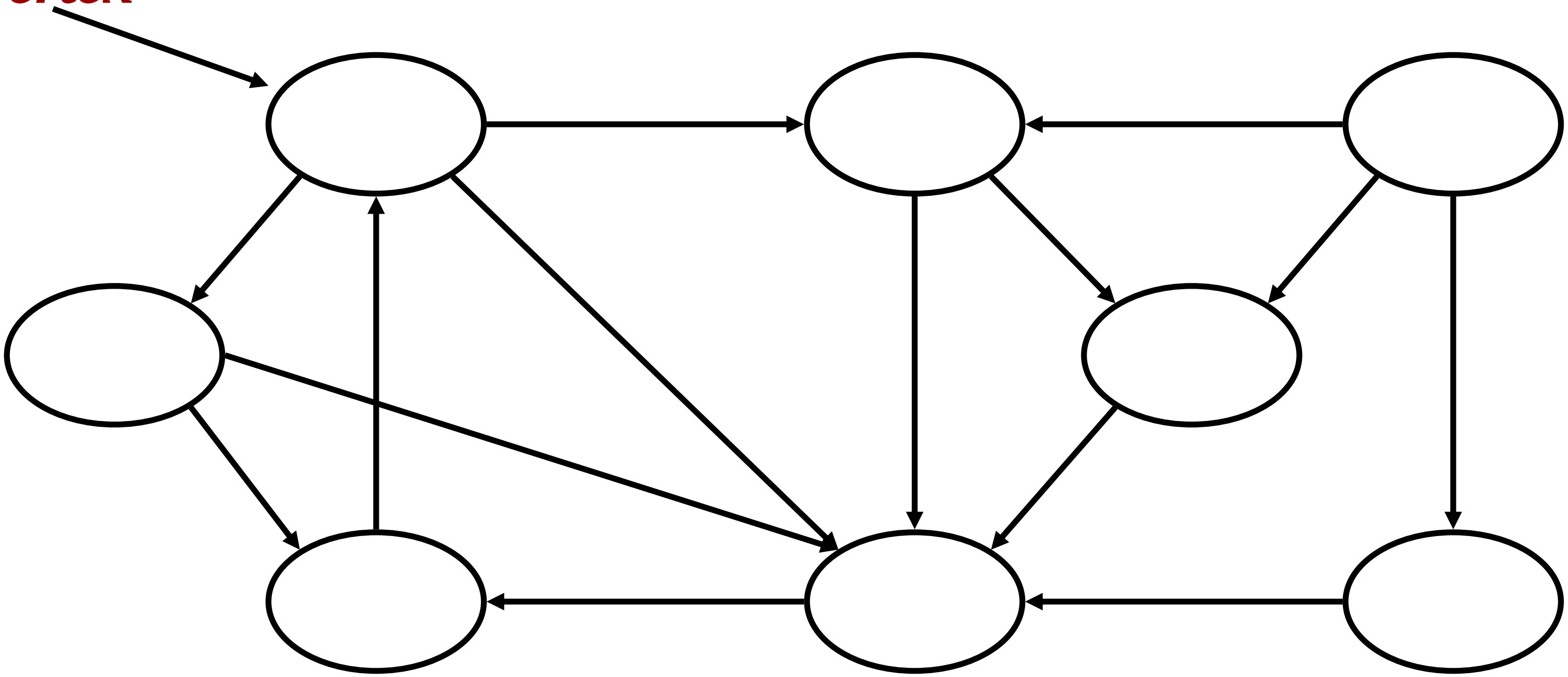
        if $mark[w] \neq visited$ **then** $dfs(w)$

Depth-first search: Directed graphs

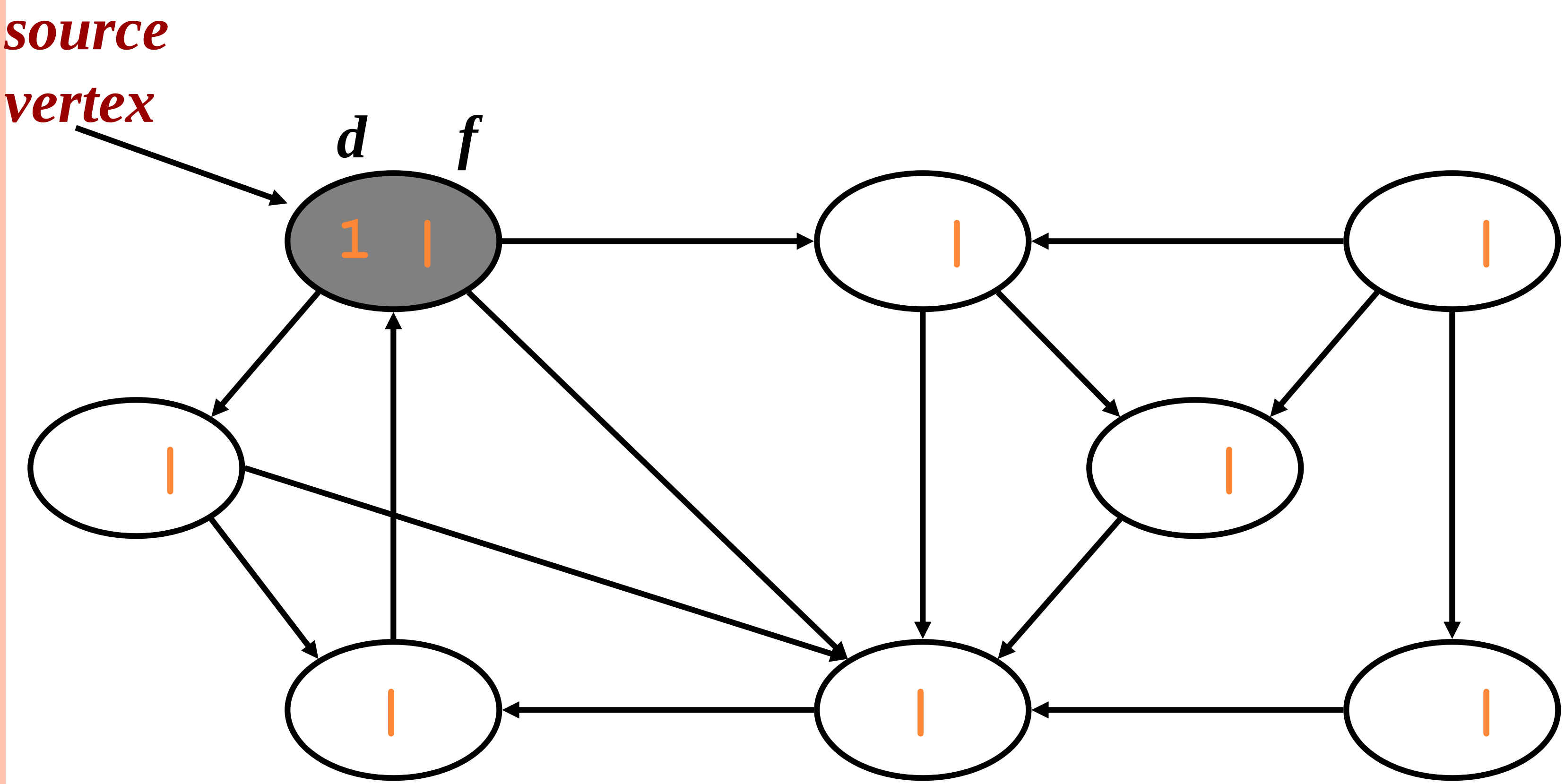
- The algorithm is essentially the same as for undirected graphs, the difference residing in the interpretation of the word "adjacent".
- In a directed graph, node **w** is adjacent to node v if the directed edge (v, w) exists.
- If **(v, w)** exists but **(w, v)** does not, then **w** is adjacent to **v** but **v** is not adjacent to **w**.
- With this change of interpretation the procedures **dfs** and **search** apply equally well in the case of a directed graph.

DFS Example

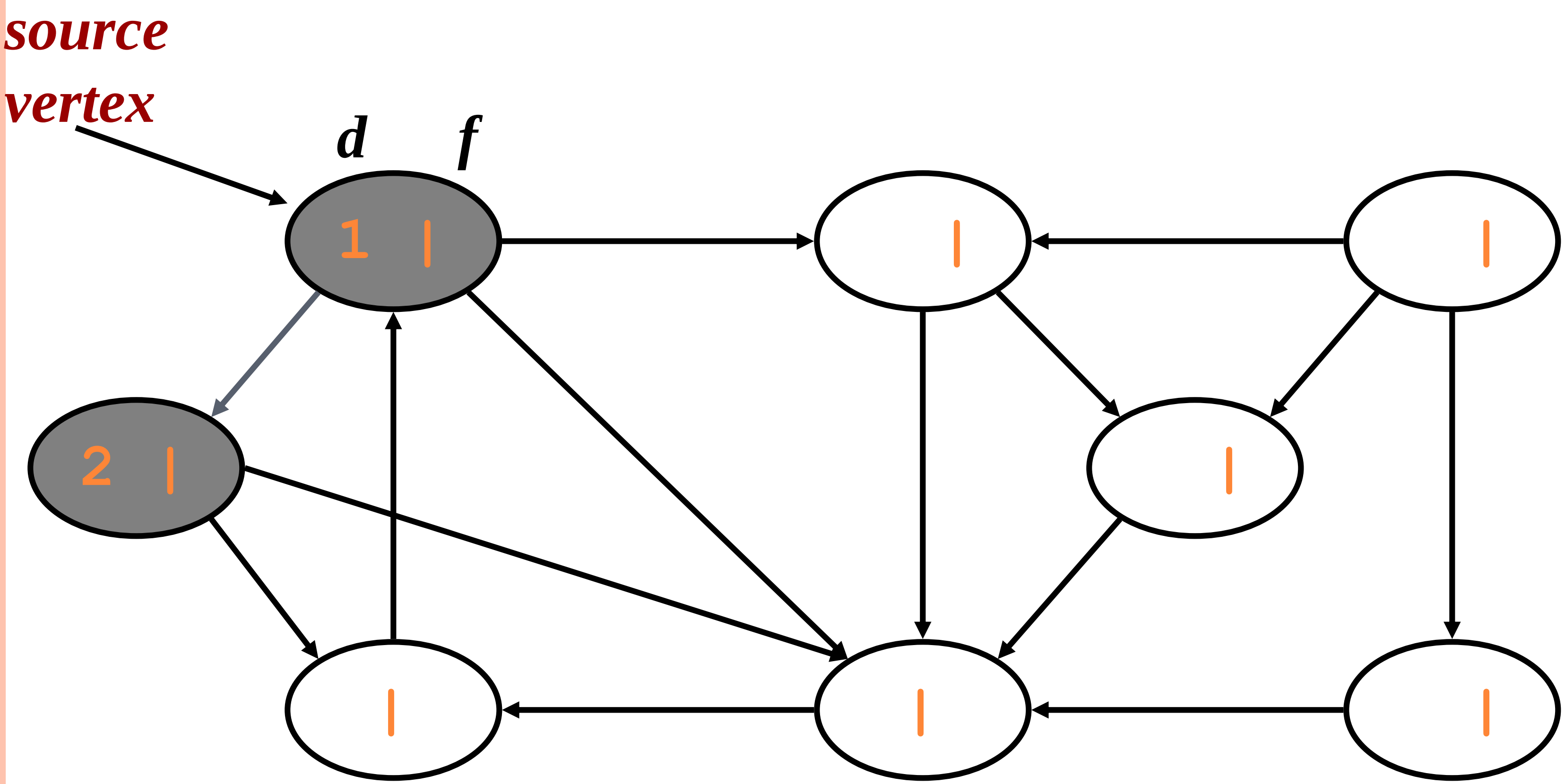
source vertex



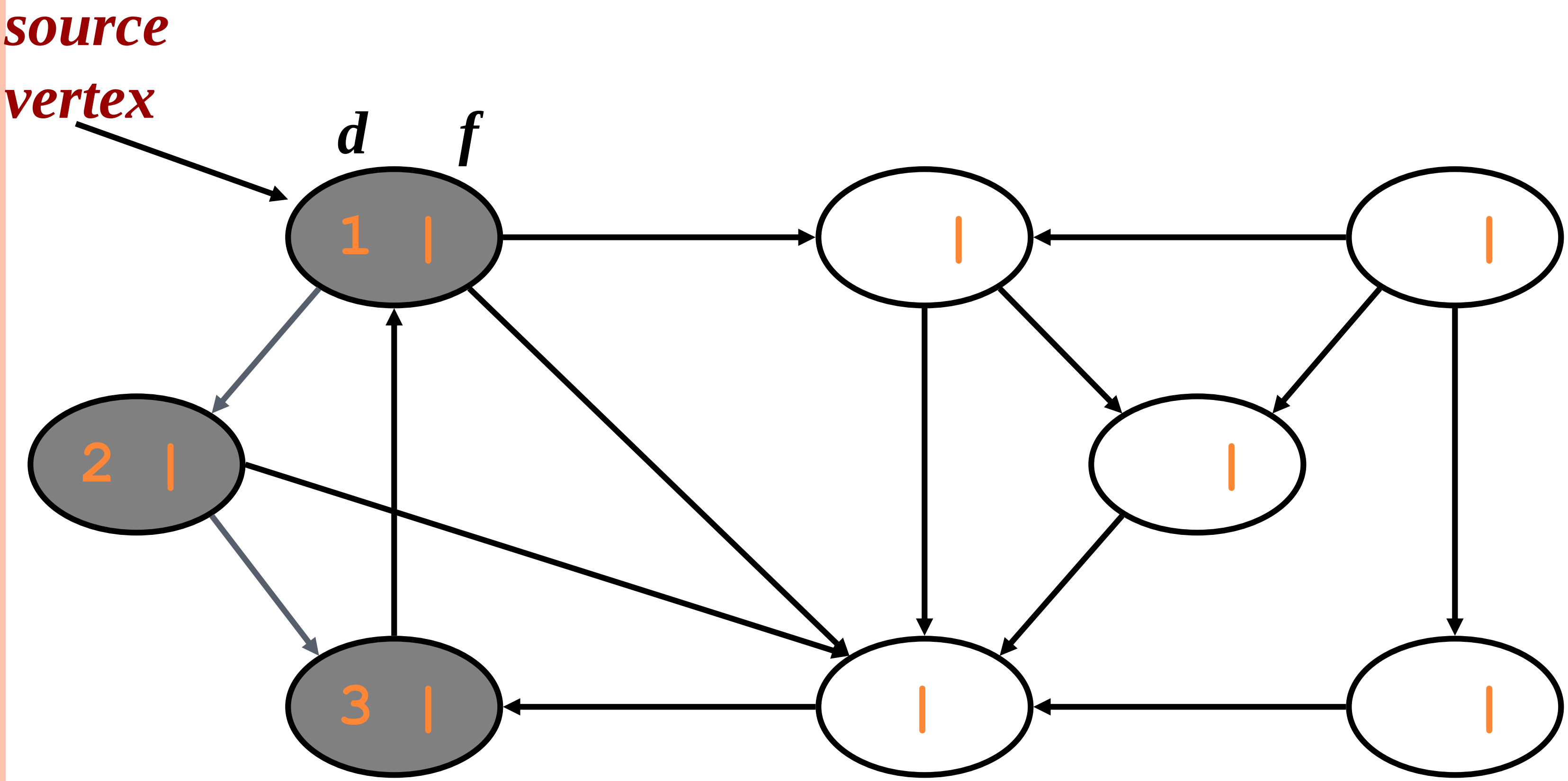
DFS Example



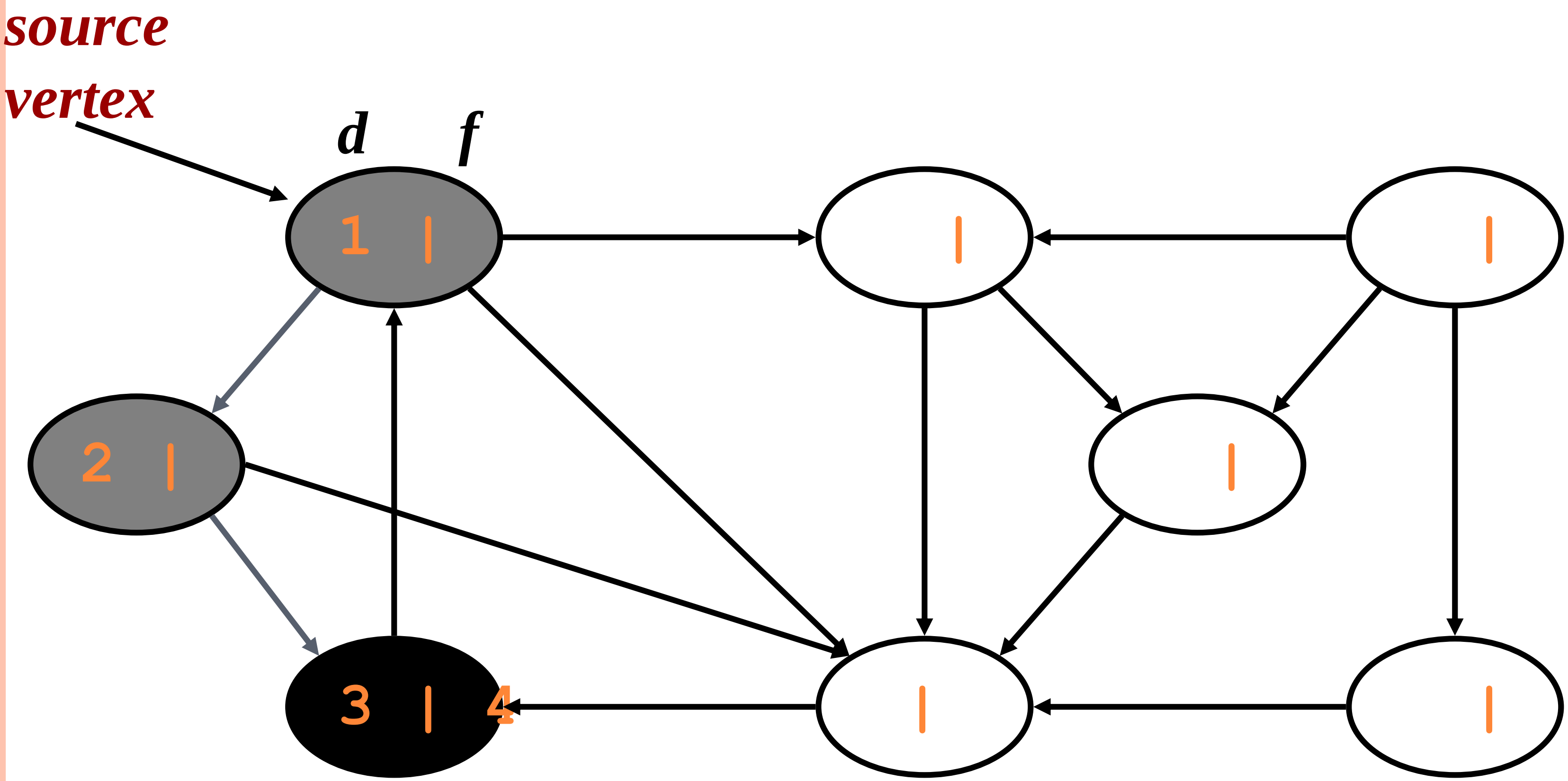
DFS Example



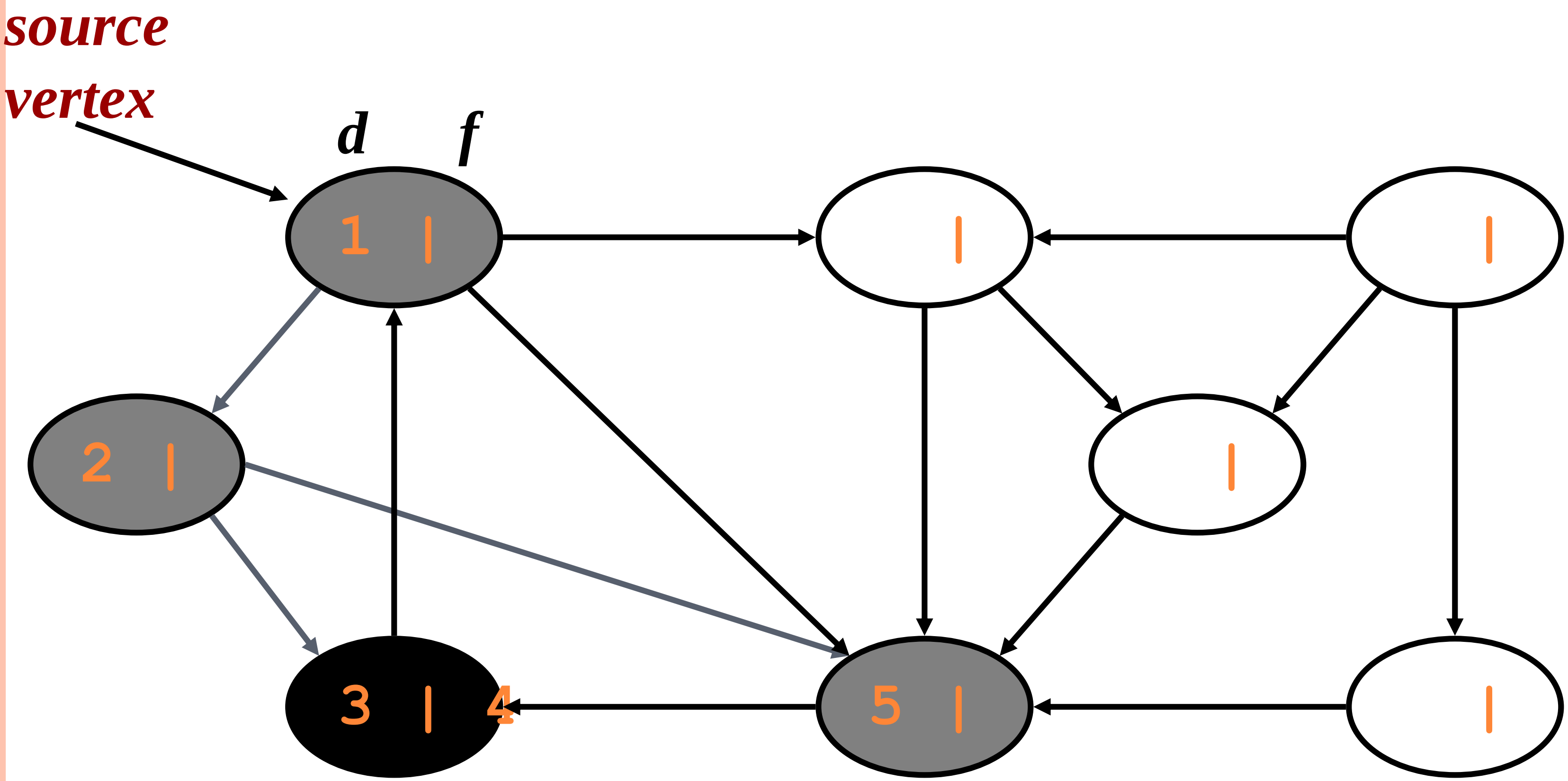
DFS EXAMPLE



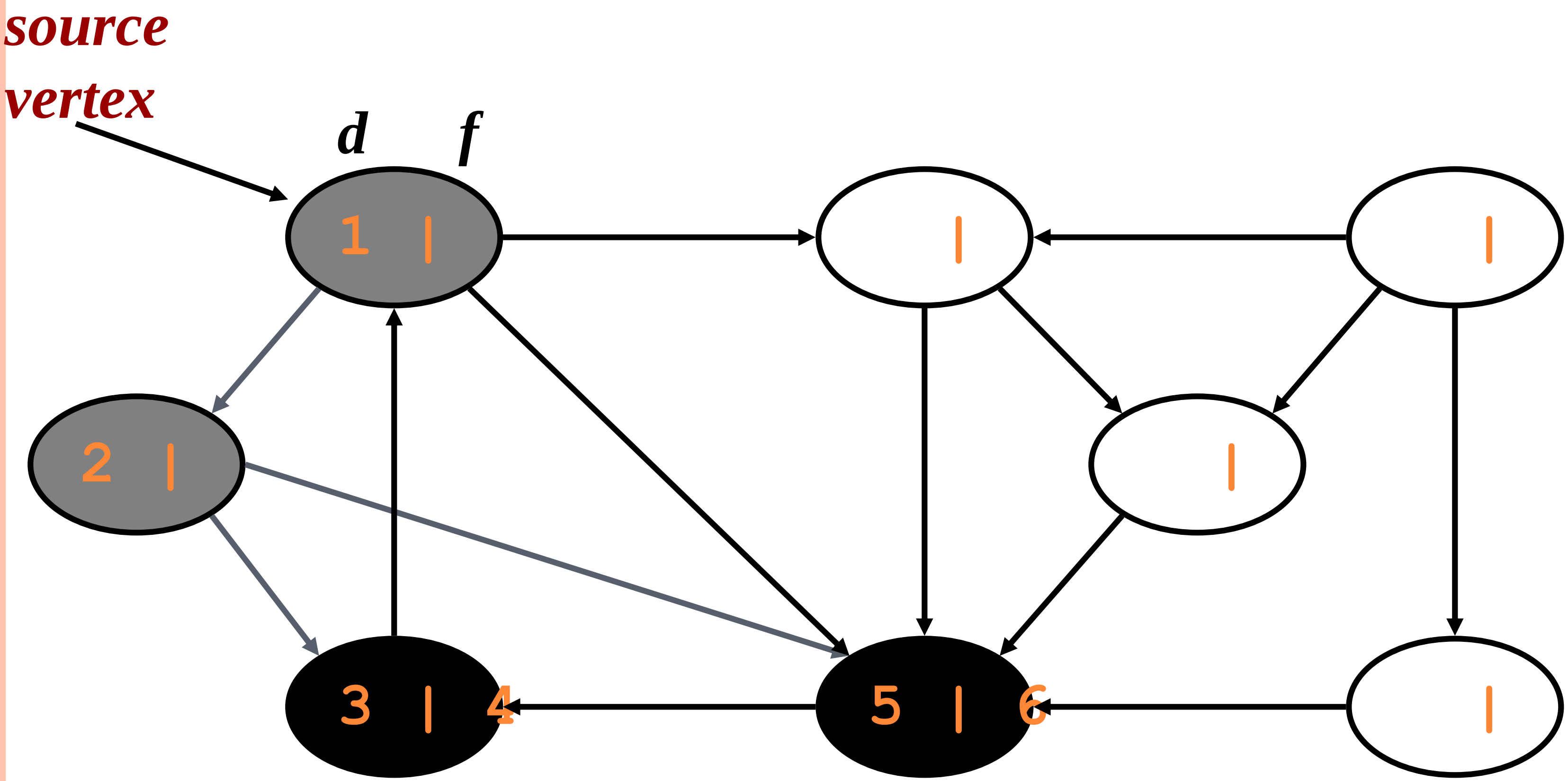
DFS Example



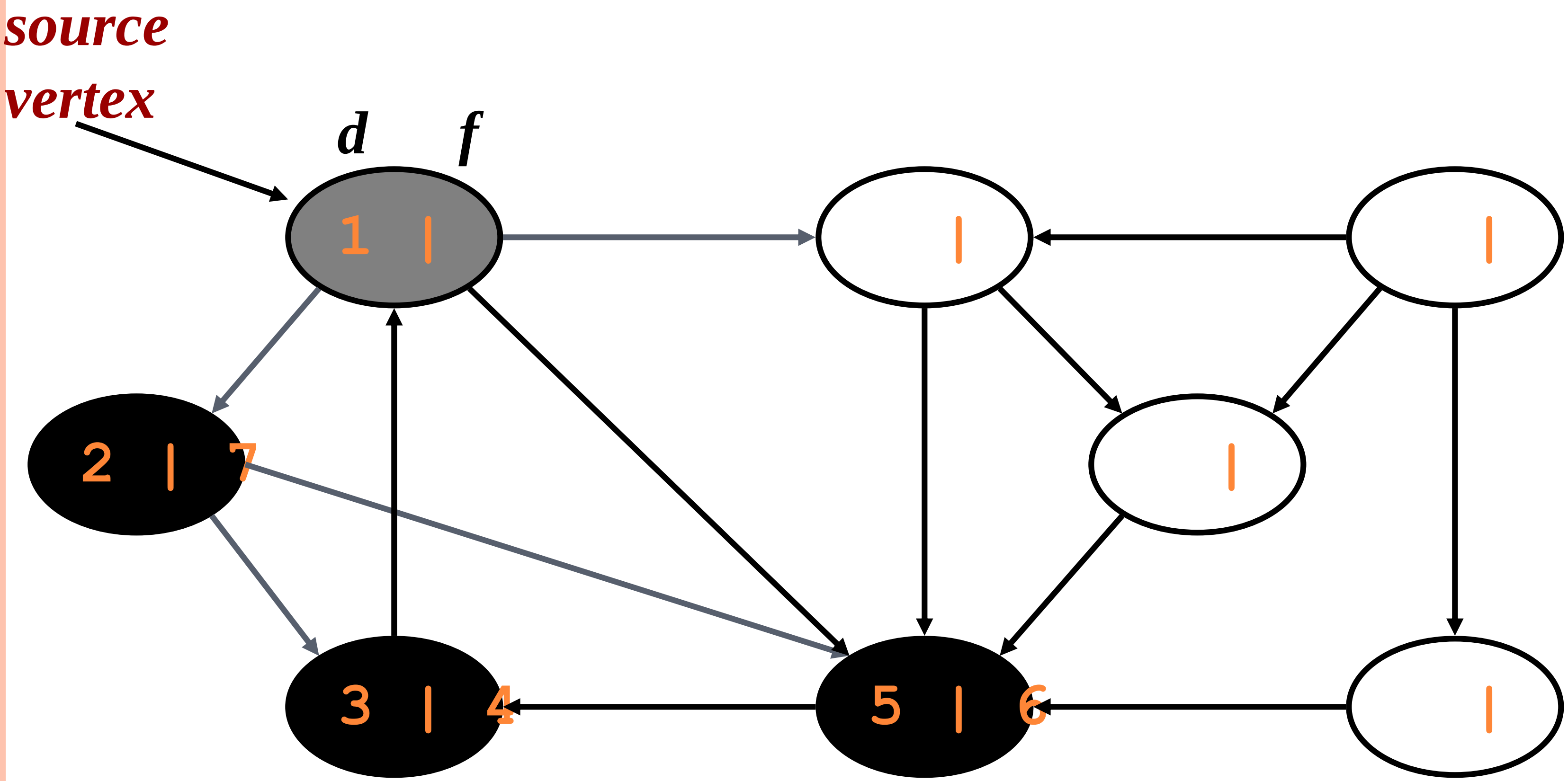
DFS Example



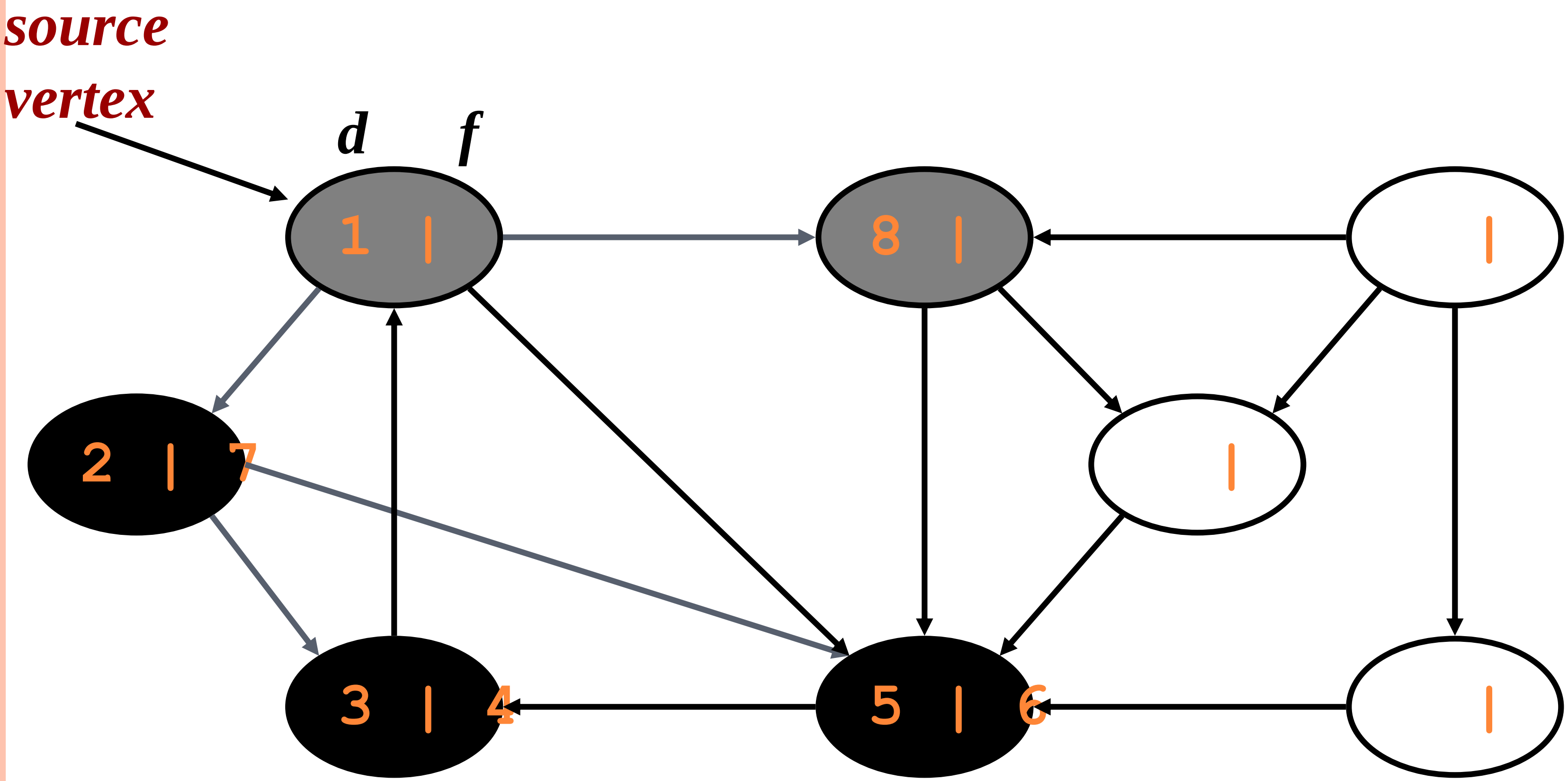
DFS Example



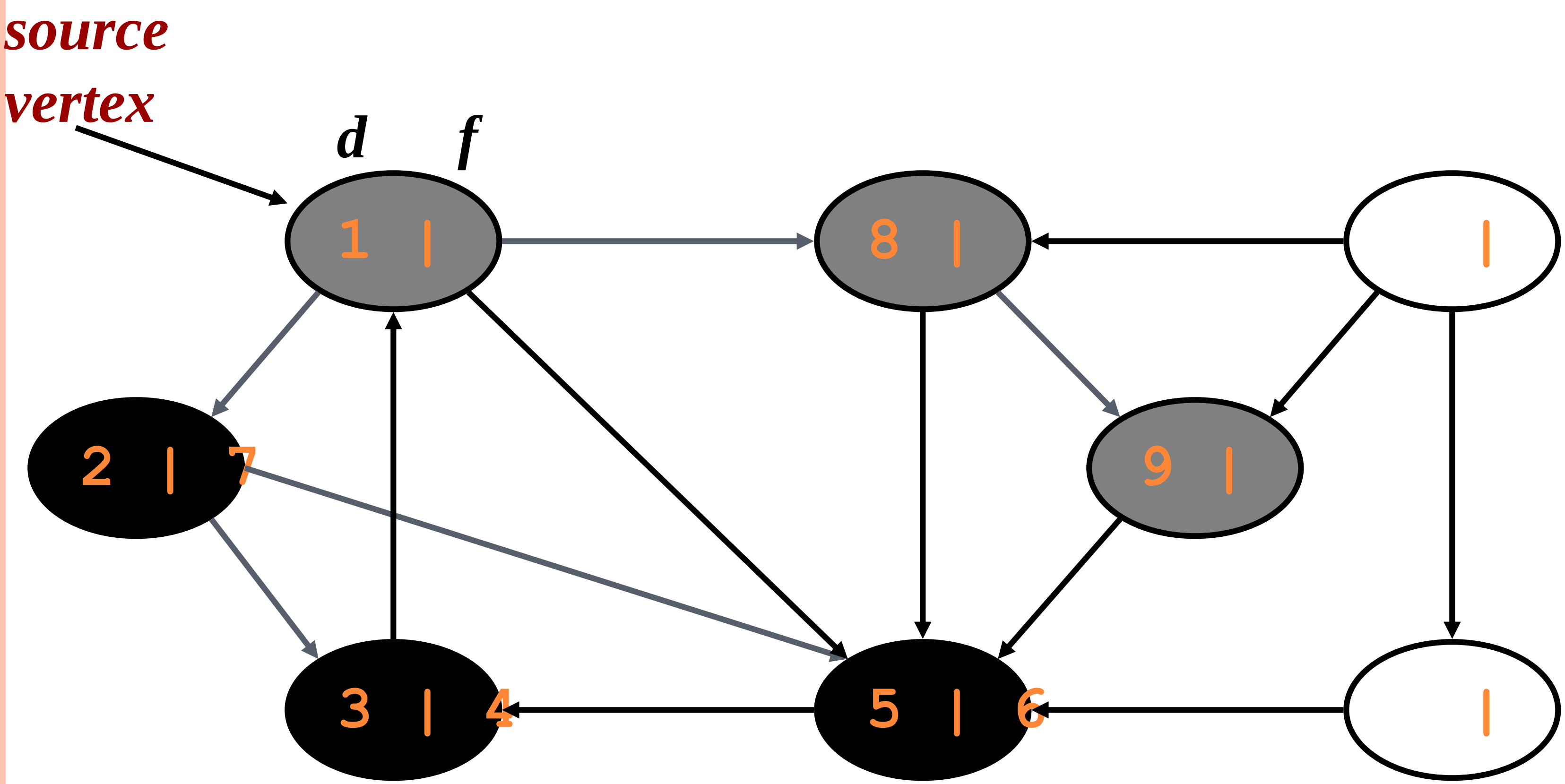
DFS Example



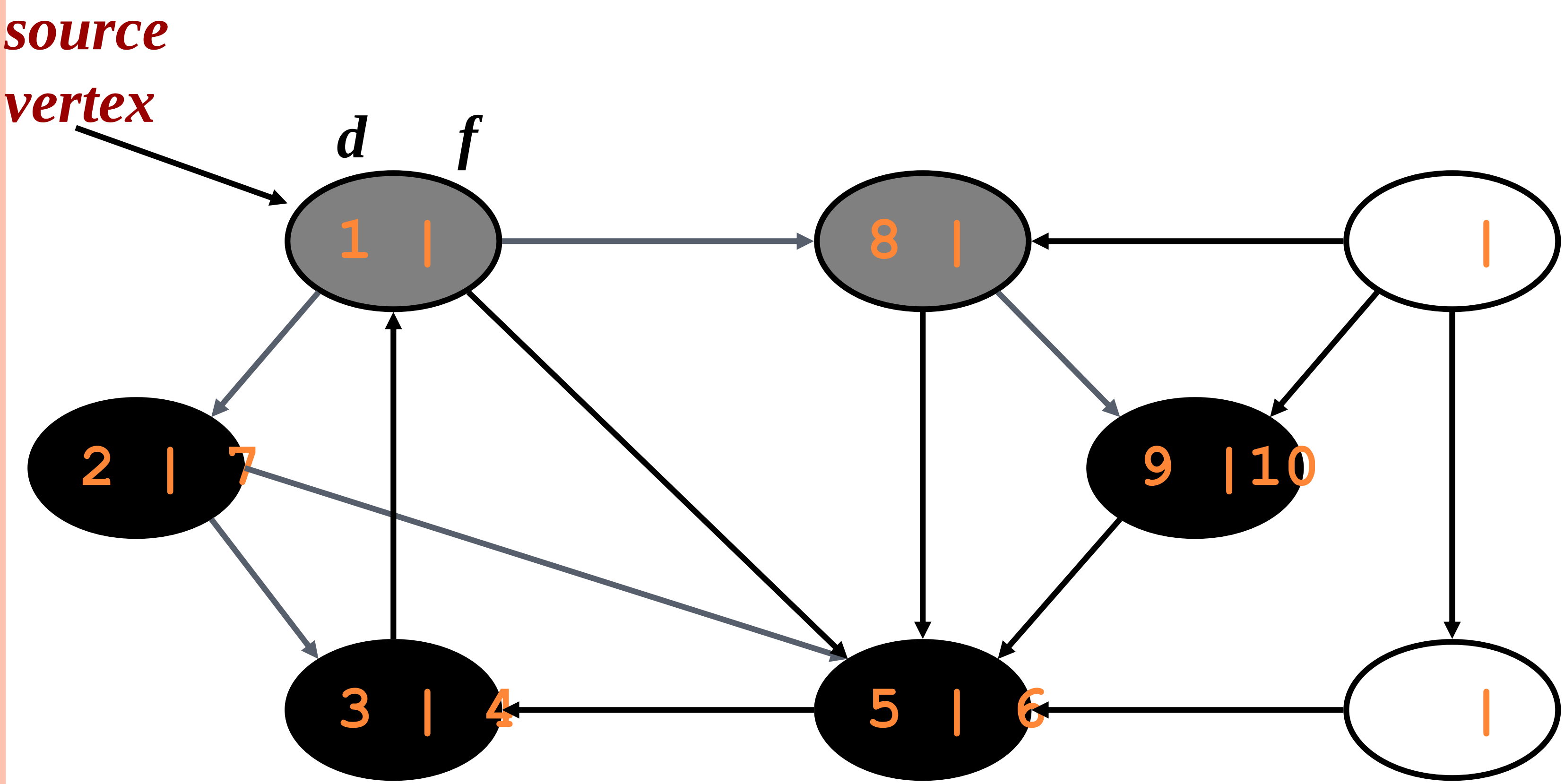
DFS Example



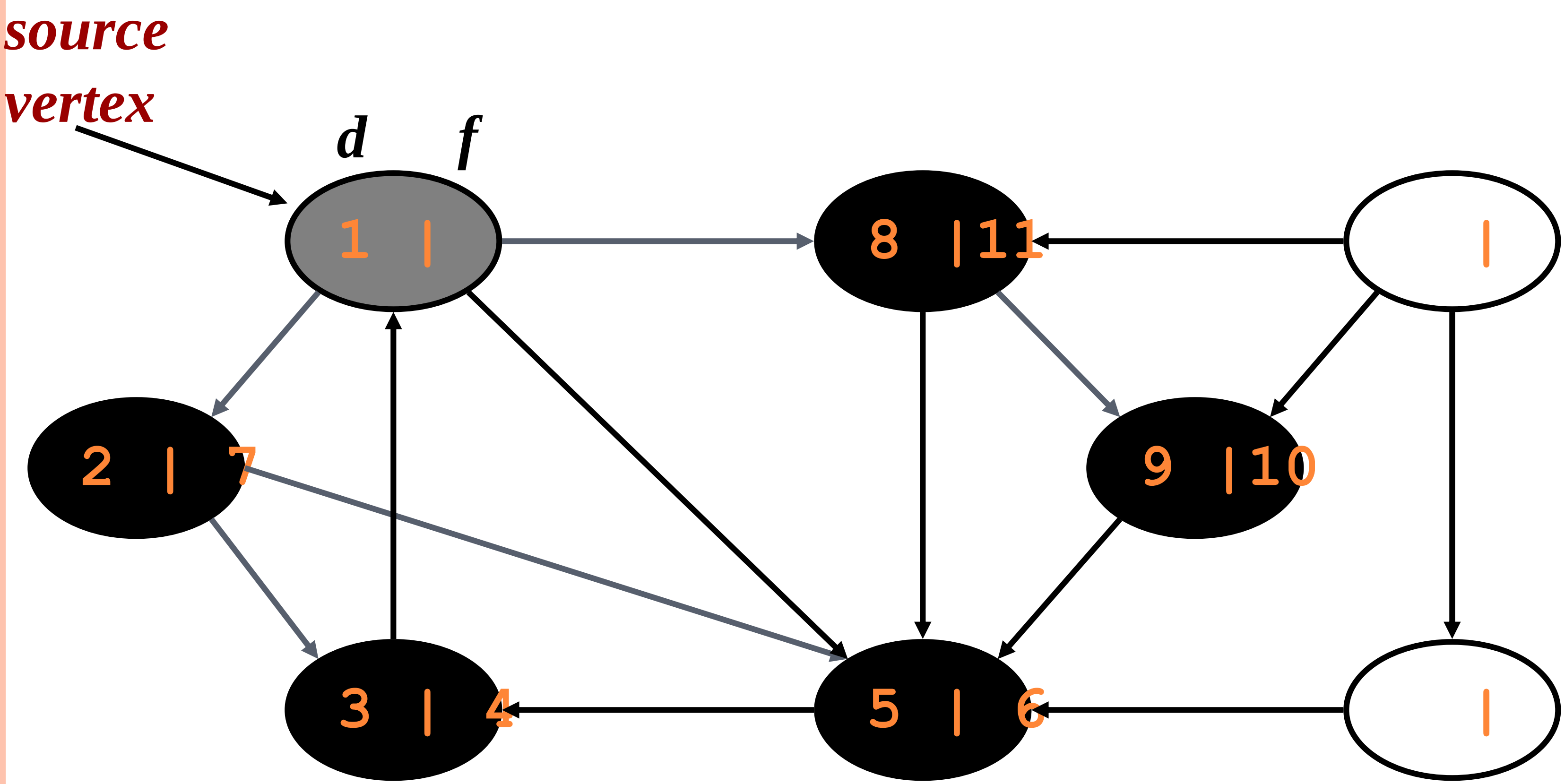
DFS Example



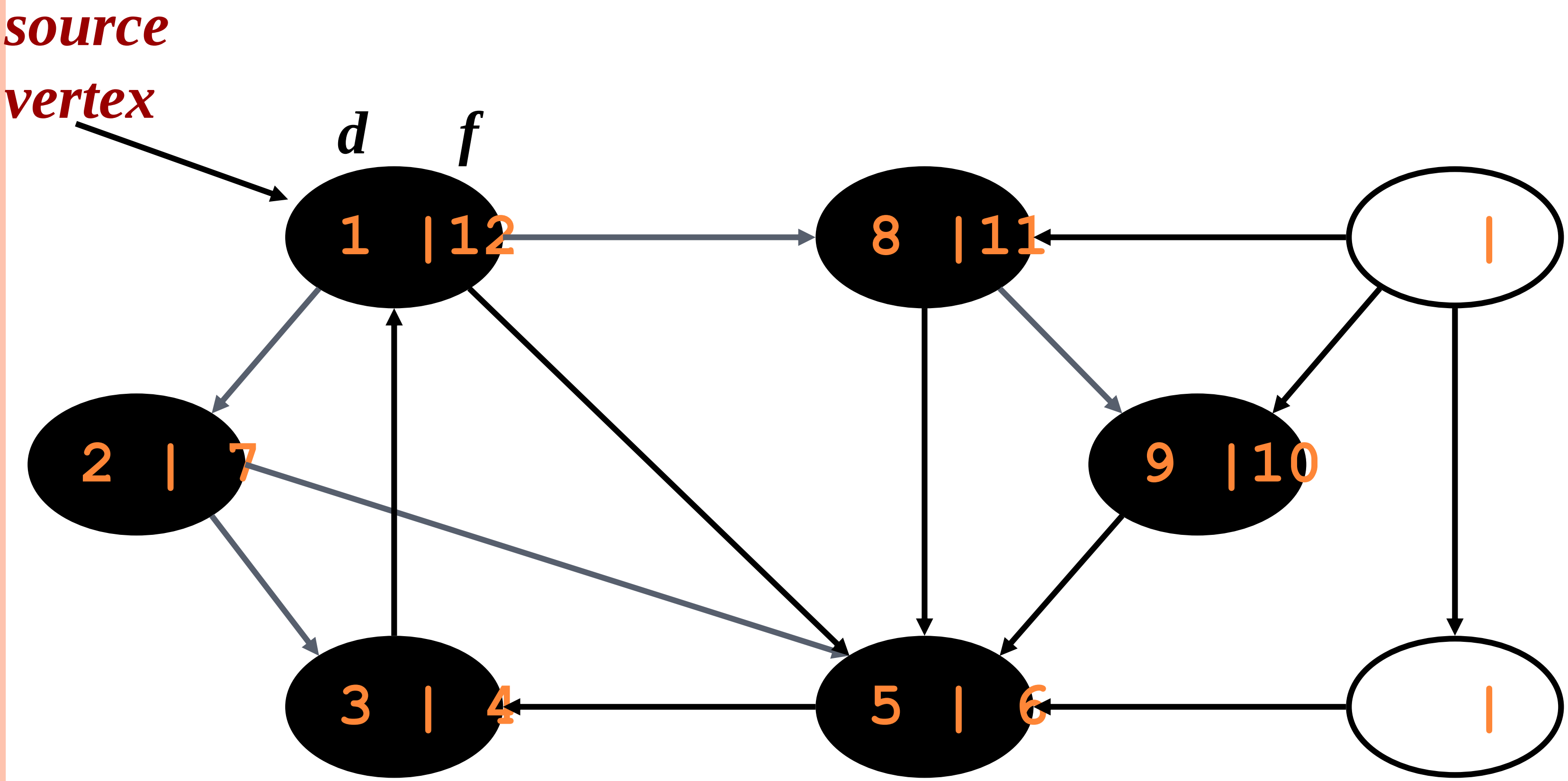
DFS Example



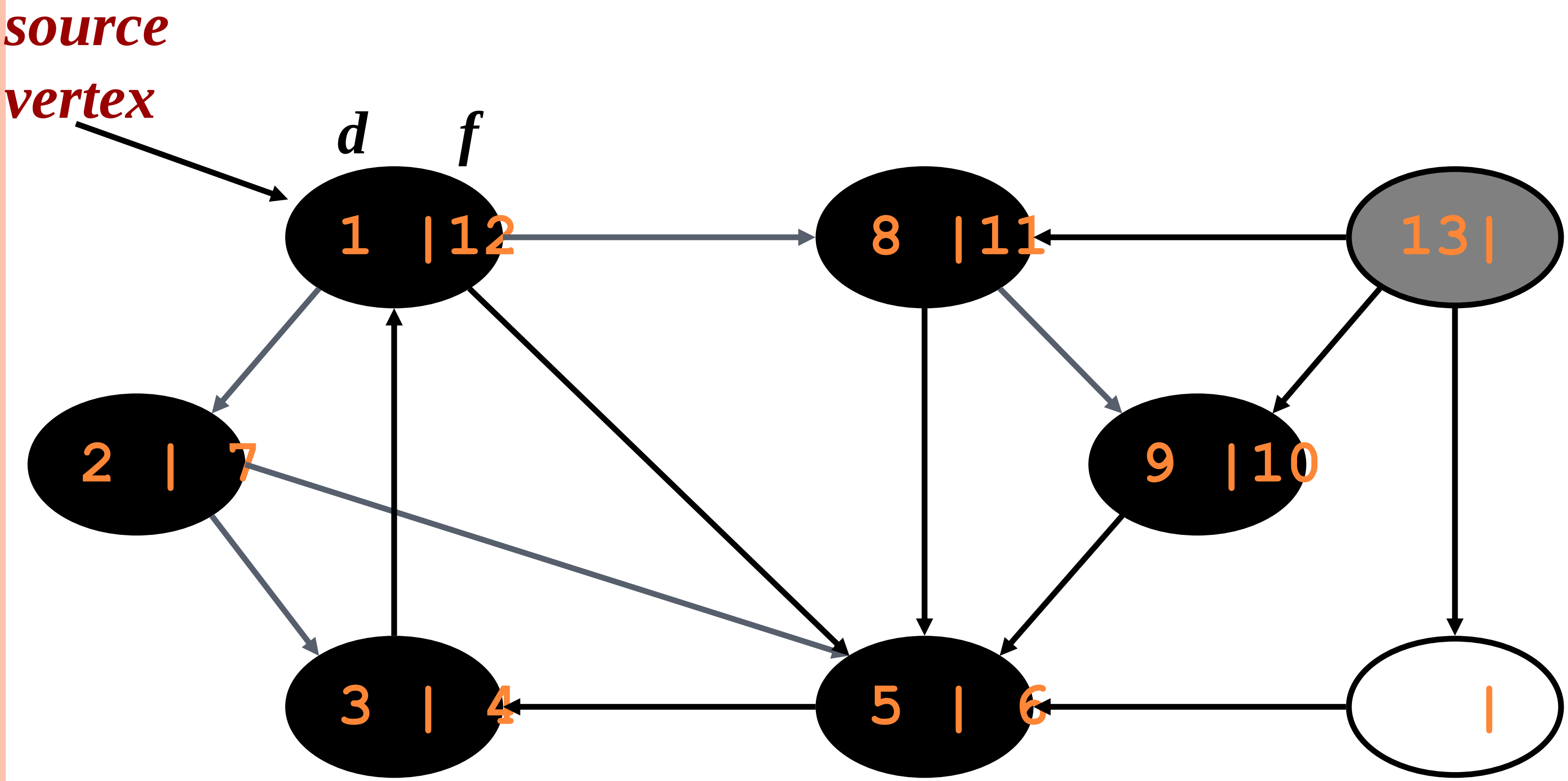
DFS Example



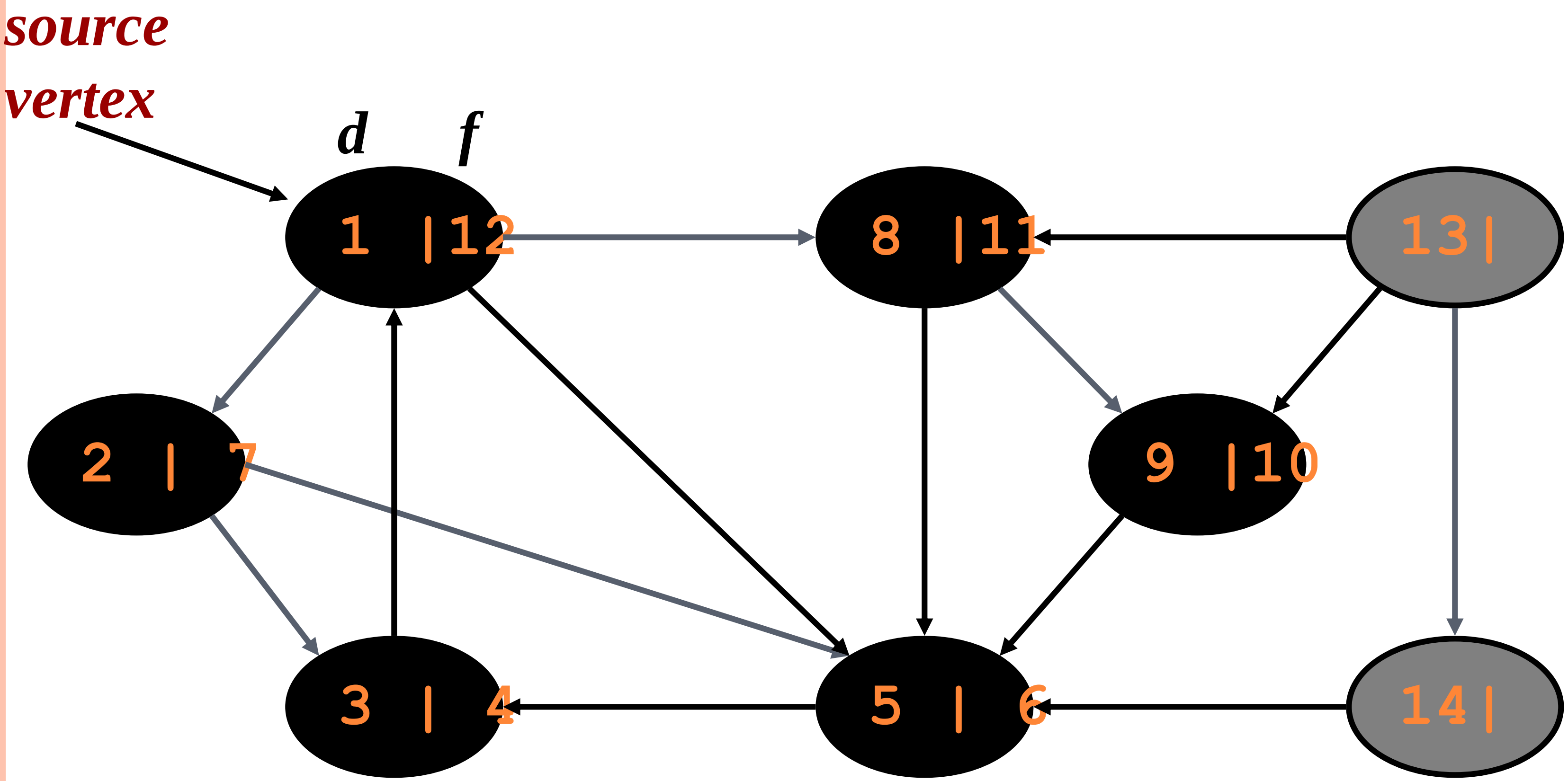
DFS Example



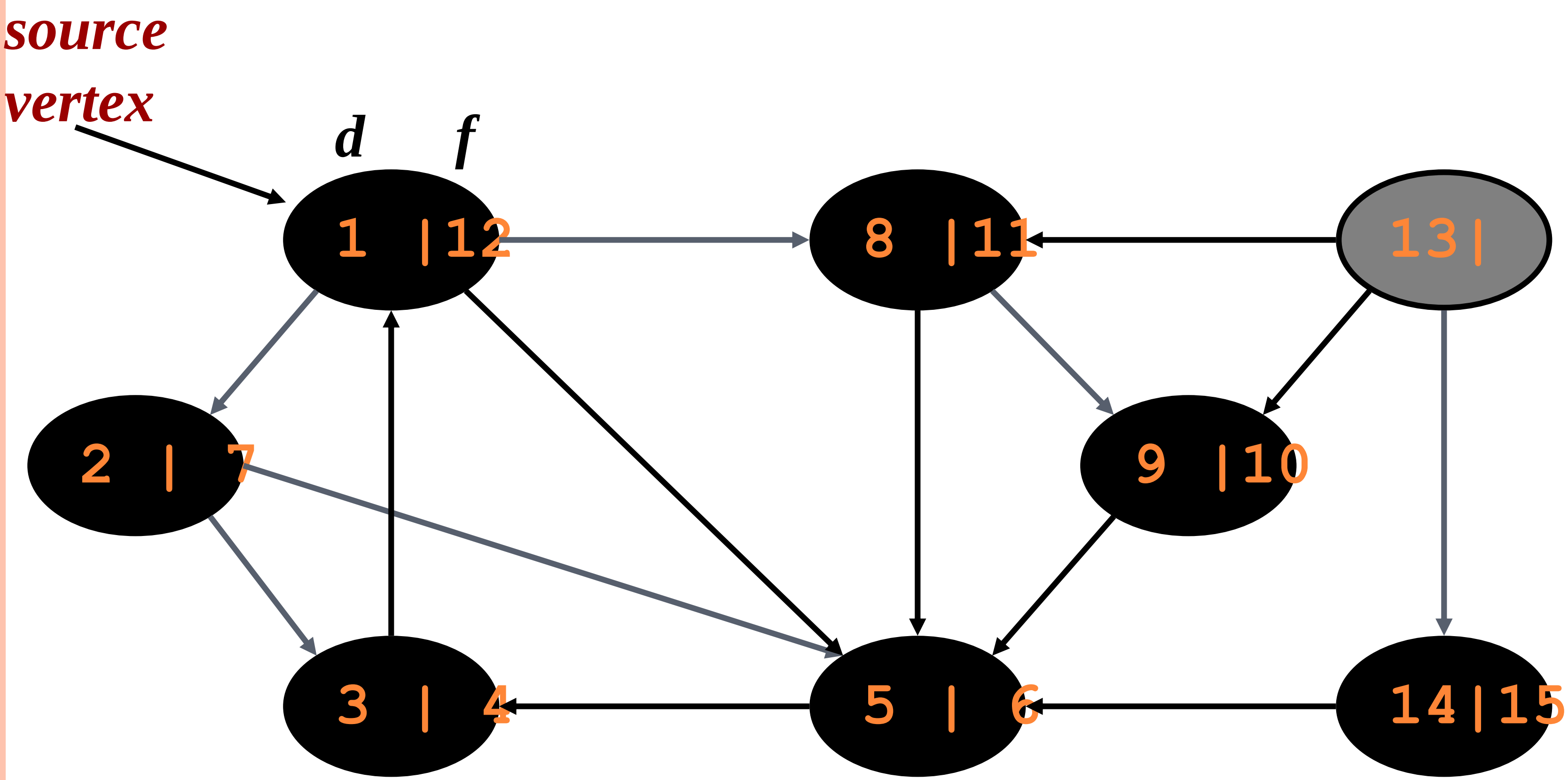
DFS Example



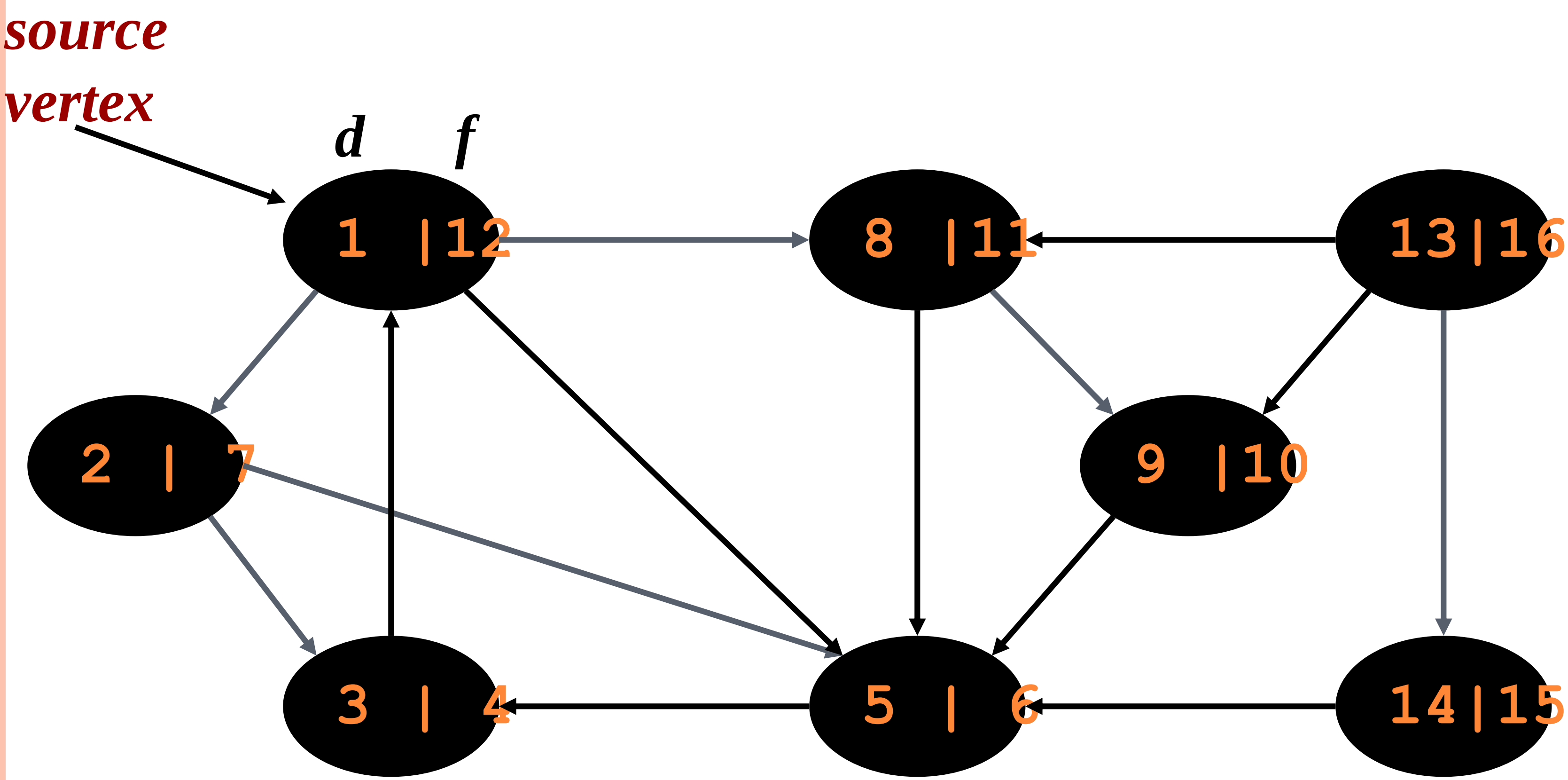
DFS Example



DFS Example



DFS EXAMPLE



Breadth-first Search (BFS)

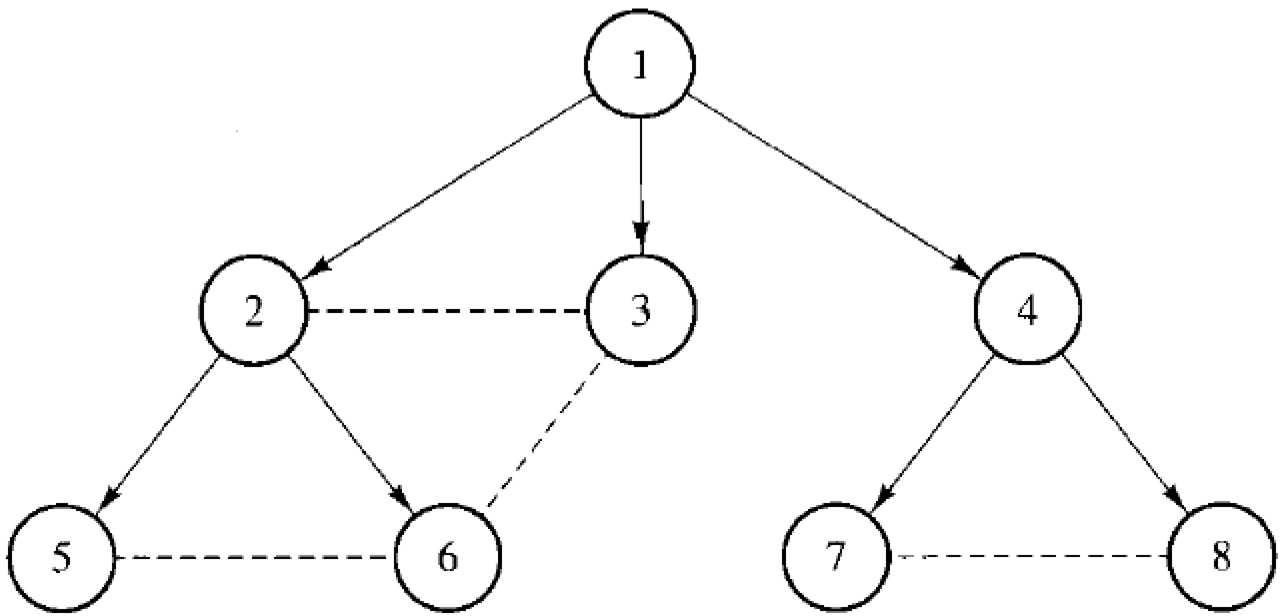
- Search for all vertices that are directly reachable from the root (called level 1 vertices)
- After mark all these vertices, visit all vertices that are directly reachable from any level 1 vertices (called level 2 vertices), and so on.
- In general, level k vertices are directly reachable from a level k – 1 vertices

```
procedure bfs(v)
    Q ← empty-queue
    mark[v] ← visited
    enqueue v into Q
    while Q is not empty do
        u ← first(Q)
        dequeue u from Q
        for each node w adjacent to u do
            if mark[w] ≠ visited then mark[w] ← visited
                enqueue w into Q
```

In both cases we need a main program to start the search.

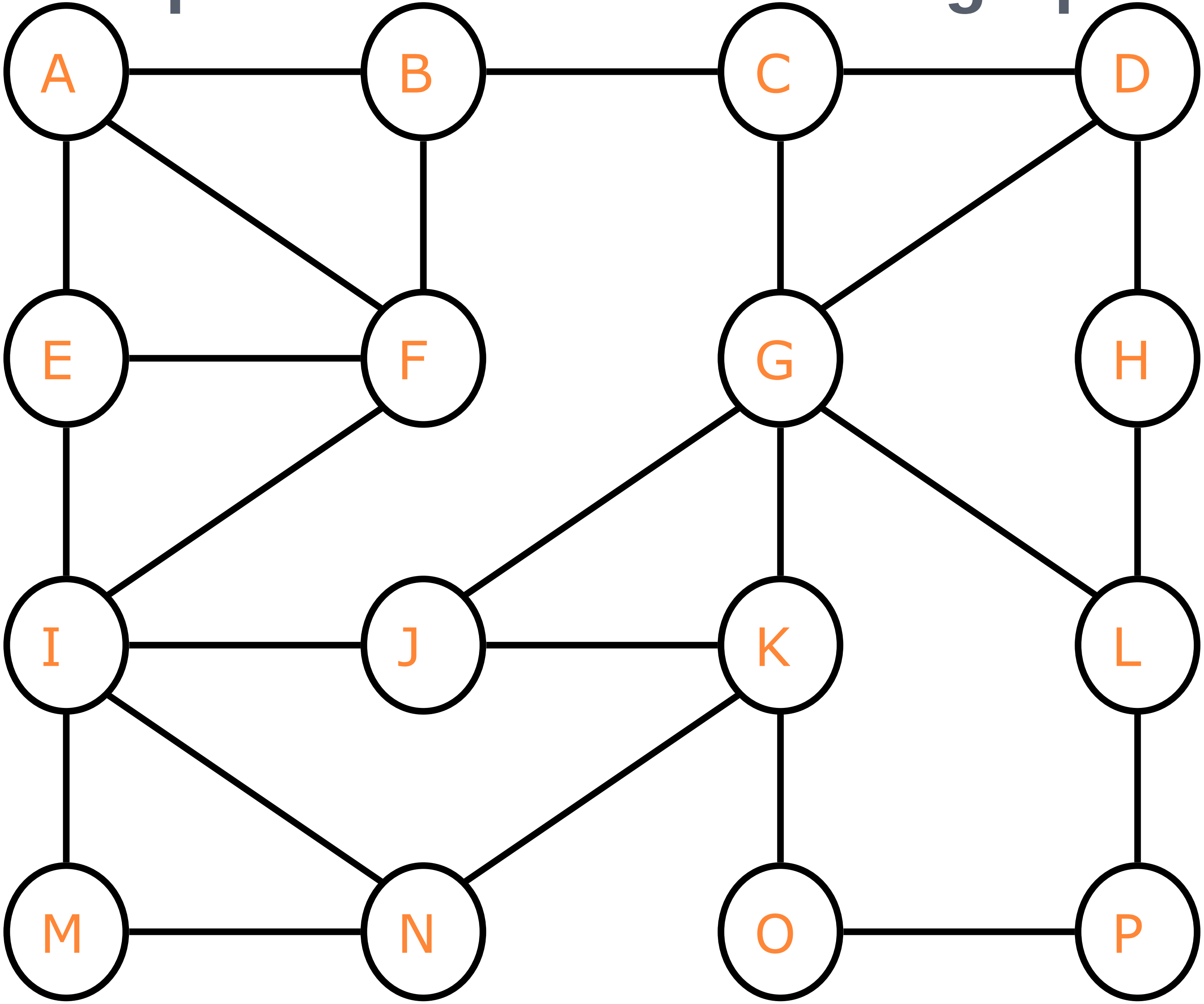
```
procedure search(G)
    for each v ∈ N do mark[v] ← not-visited
    for each v ∈ N do
        if mark[v] ≠ visited then {dfs2 or bfs} (v)
```

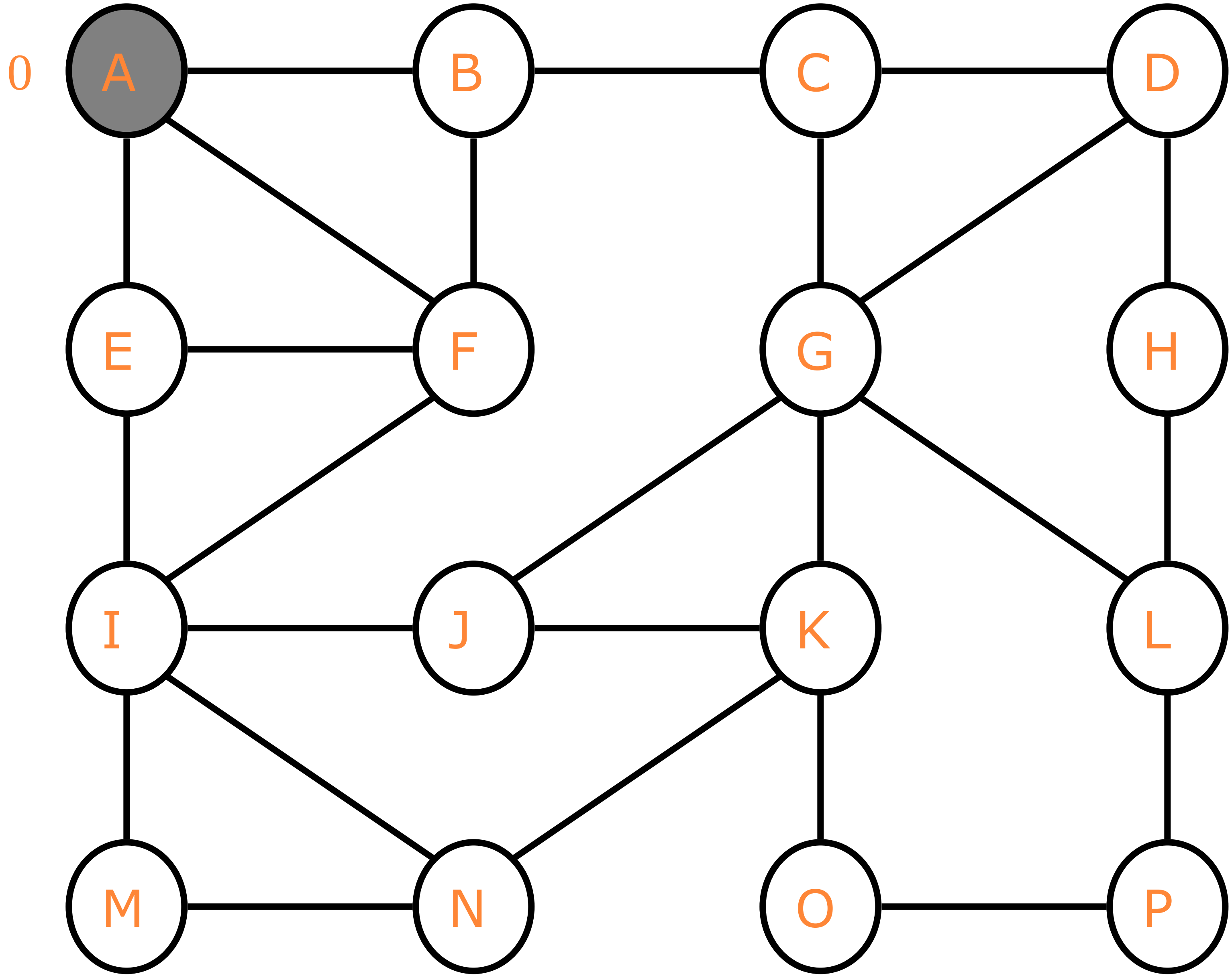
EXAMPLE:BFS FOR DIRECTED GRAPH

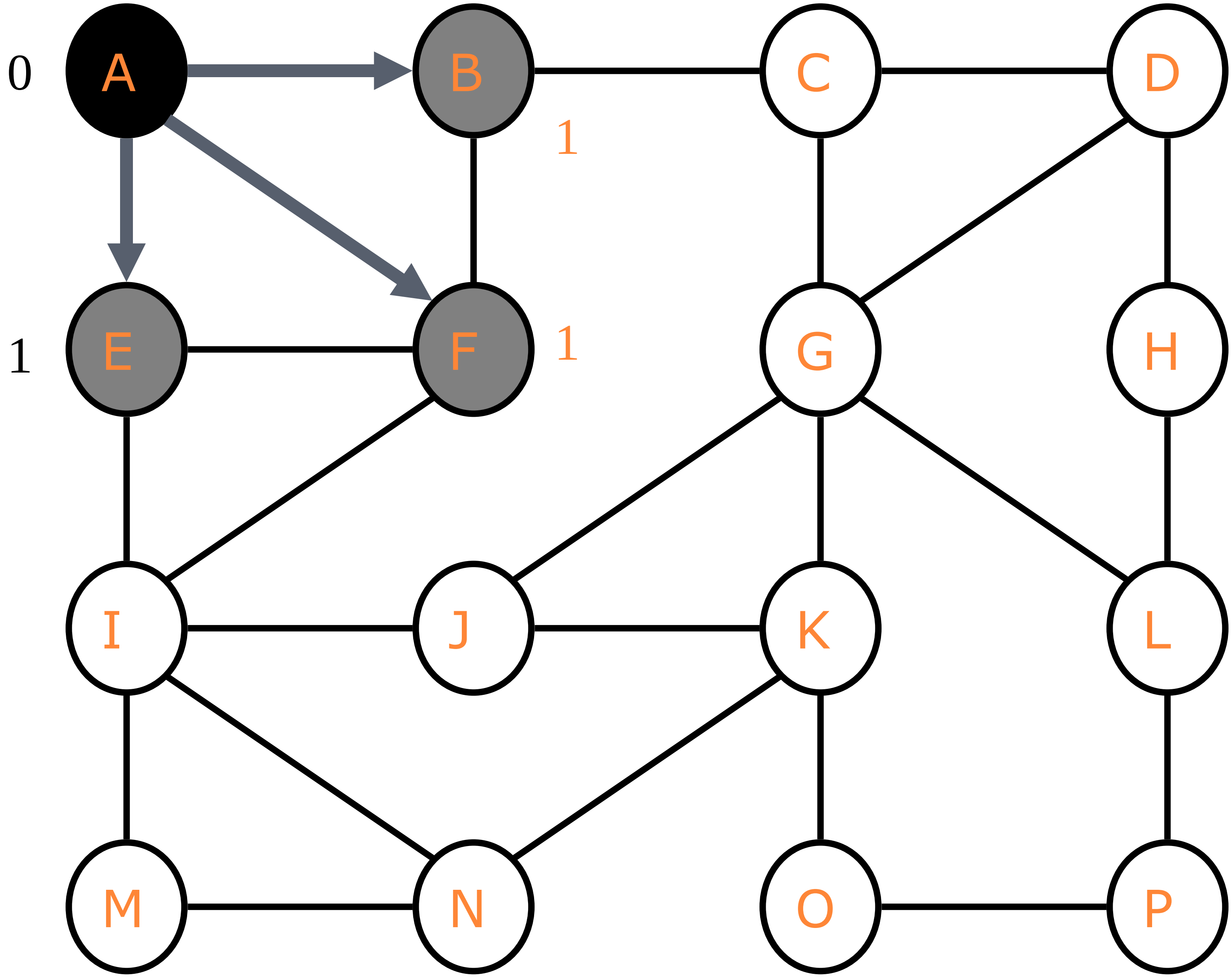


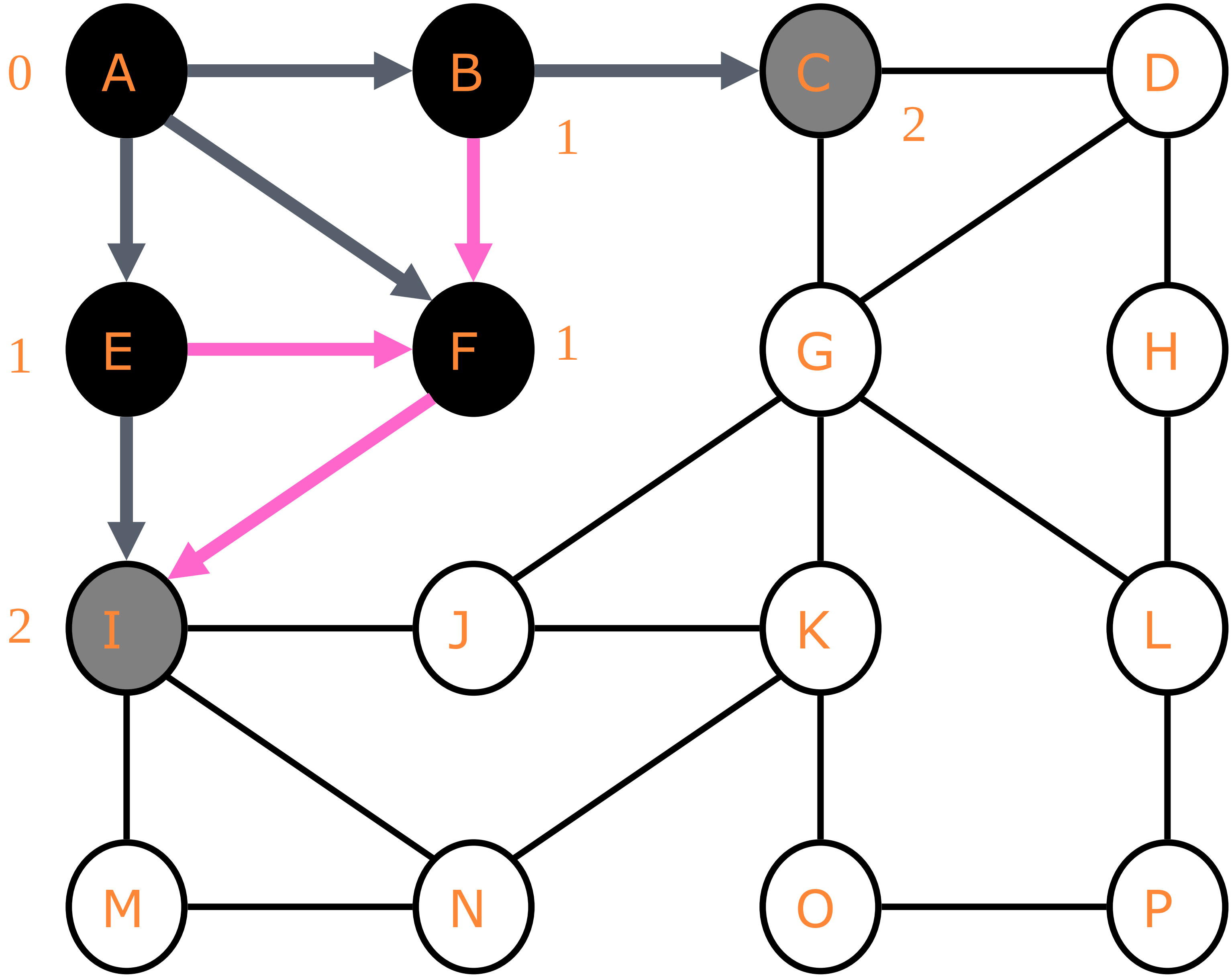
	Node visited	Q
1.	1	2,3,4
2.	2	3,4,5,6
3.	3	4,5,6
4.	4	5,6,7,8
5.	5	6,7,8
6.	6	7,8
7.	7	8
8.	8	—

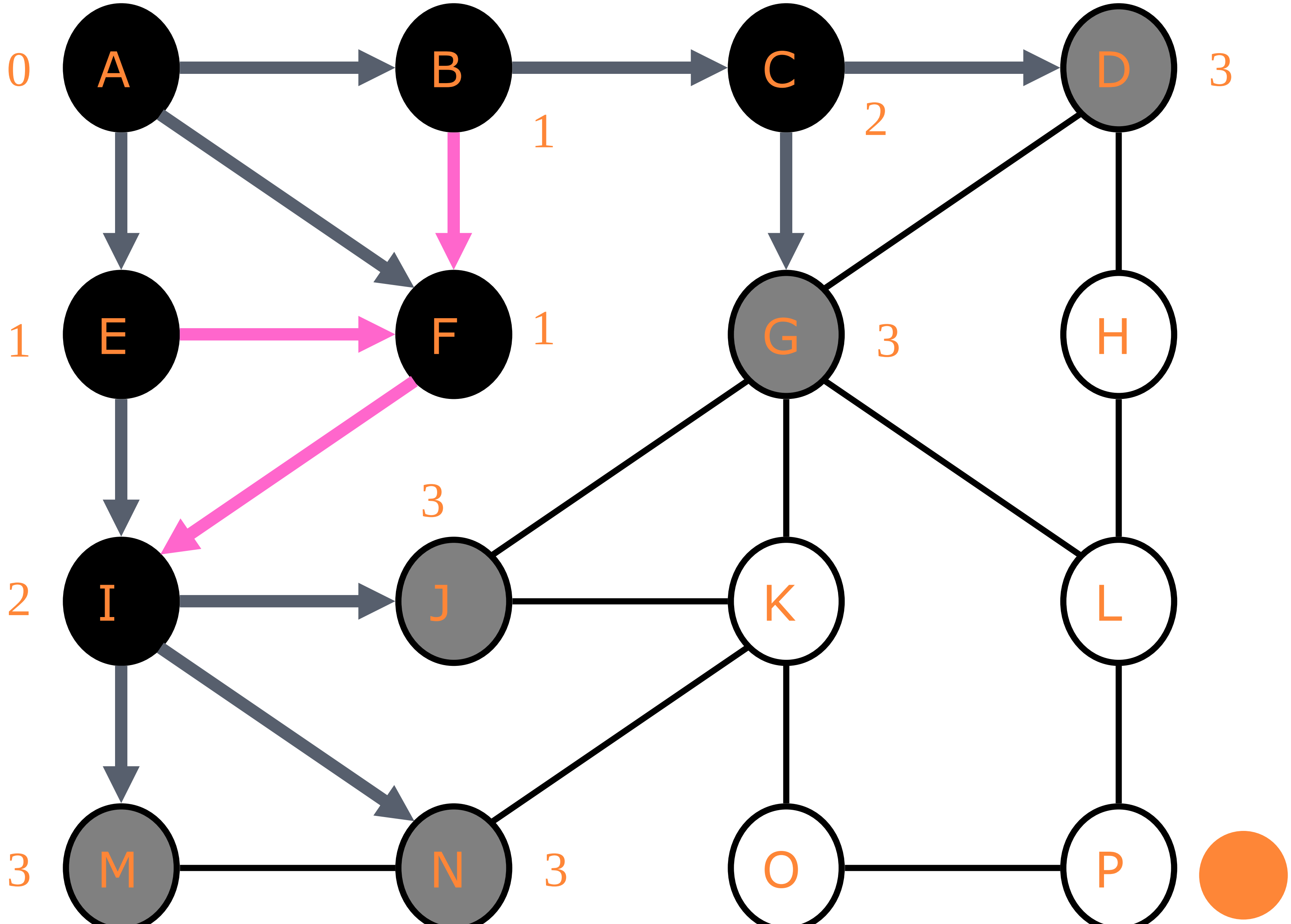
An Example:BFS for undirected graph

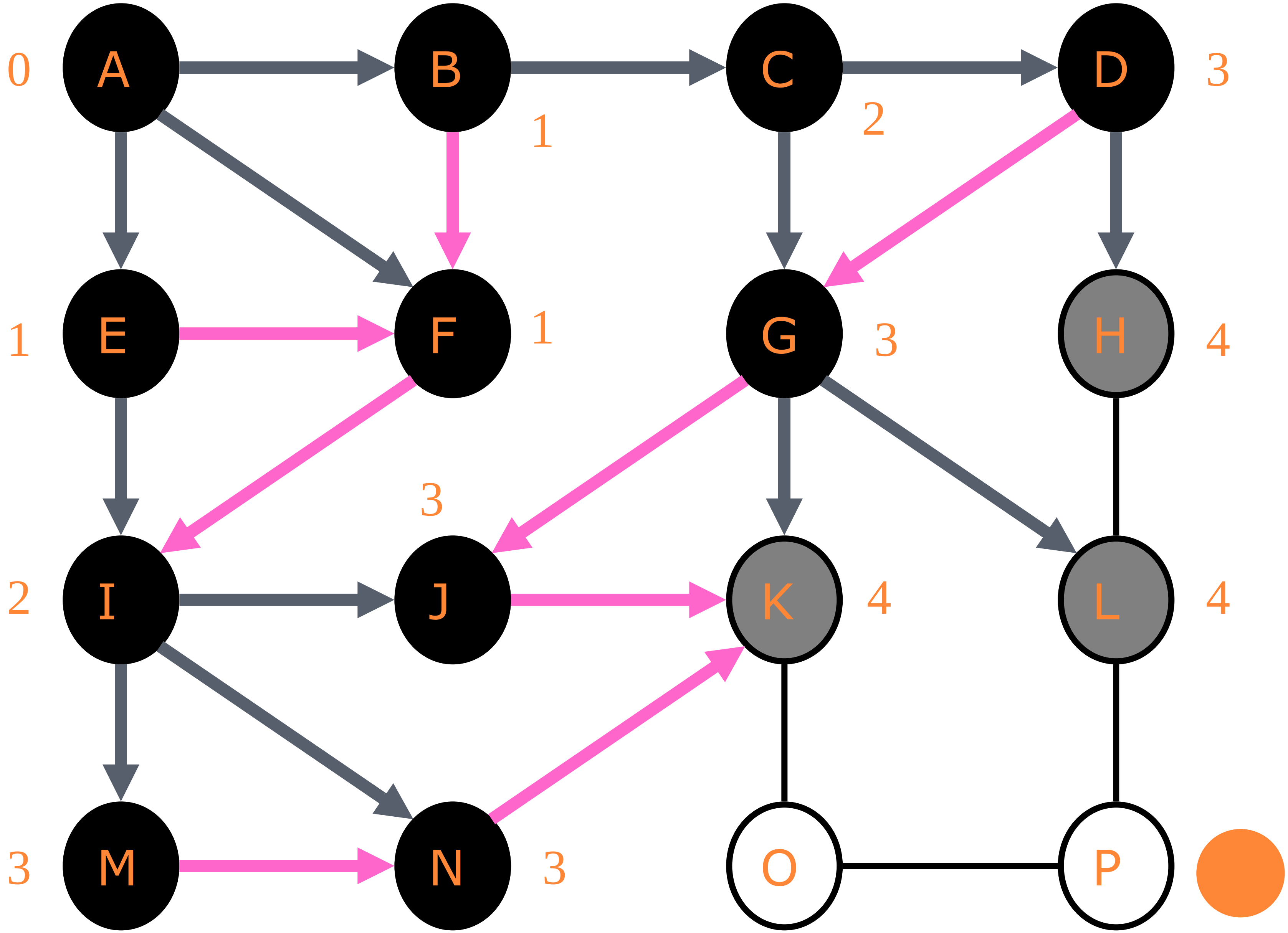


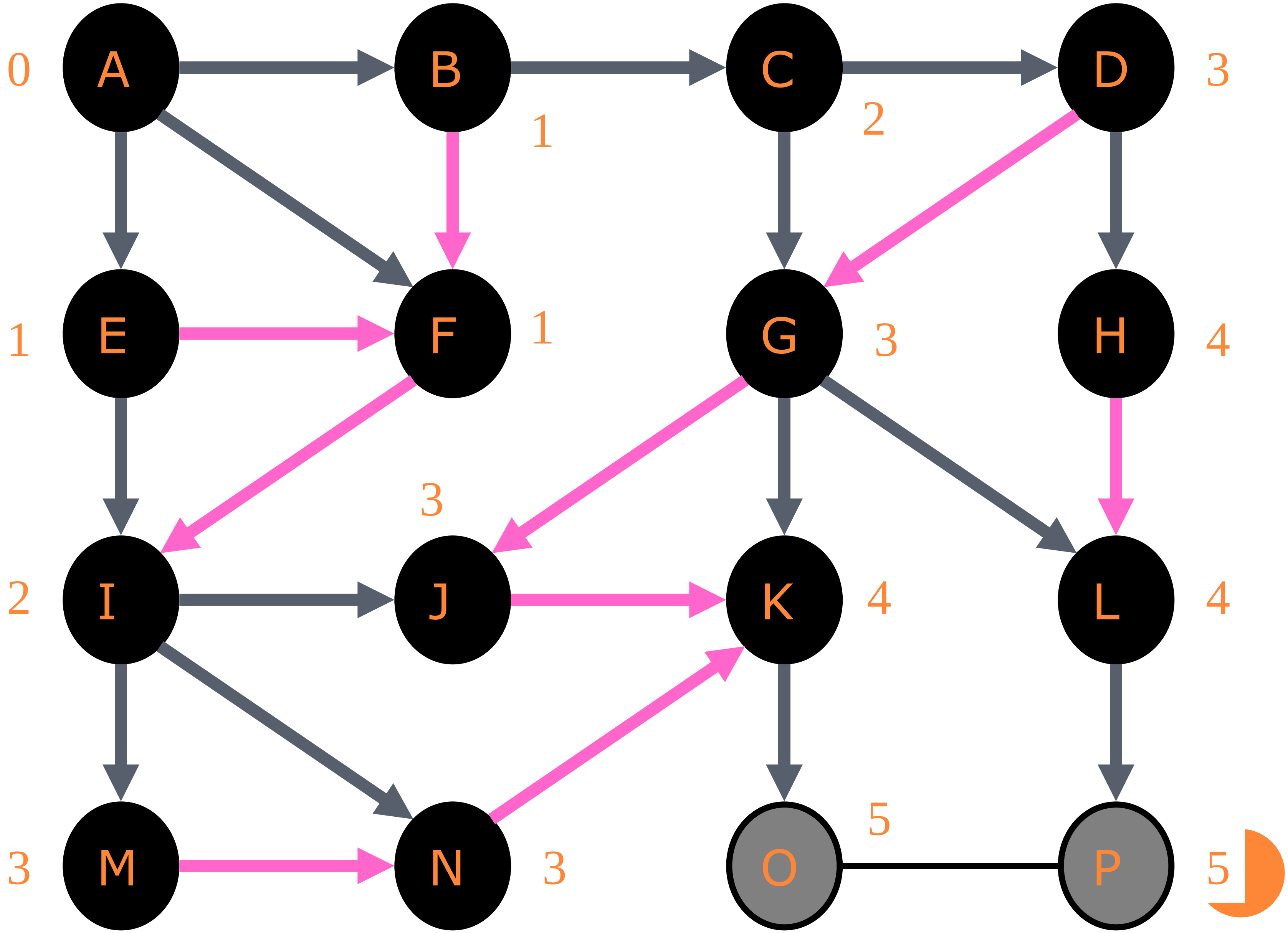

0
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

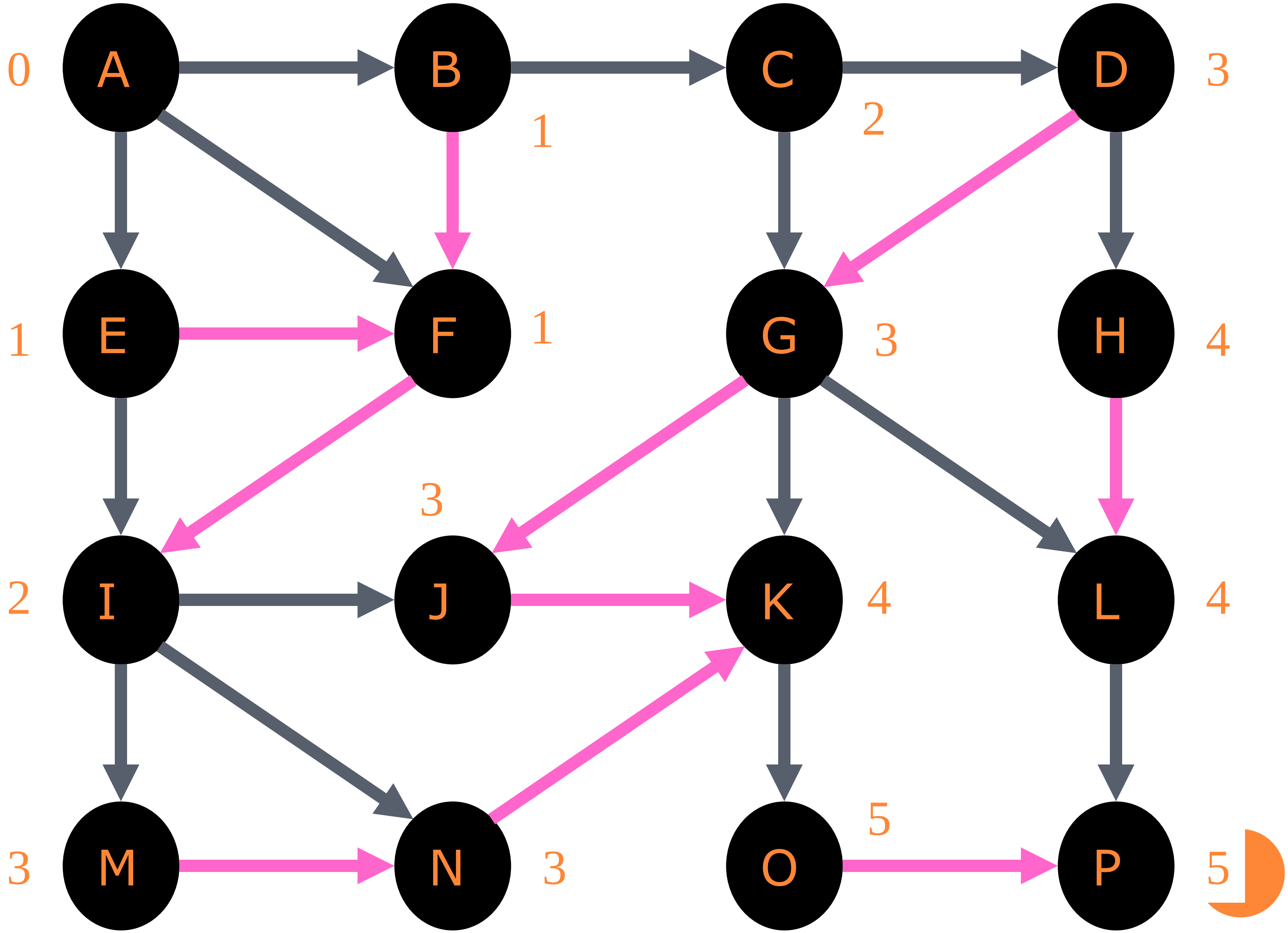

0
A
B
1
C
D
1
E
F
1
G
H
I
J
K
L
M
N
O
P

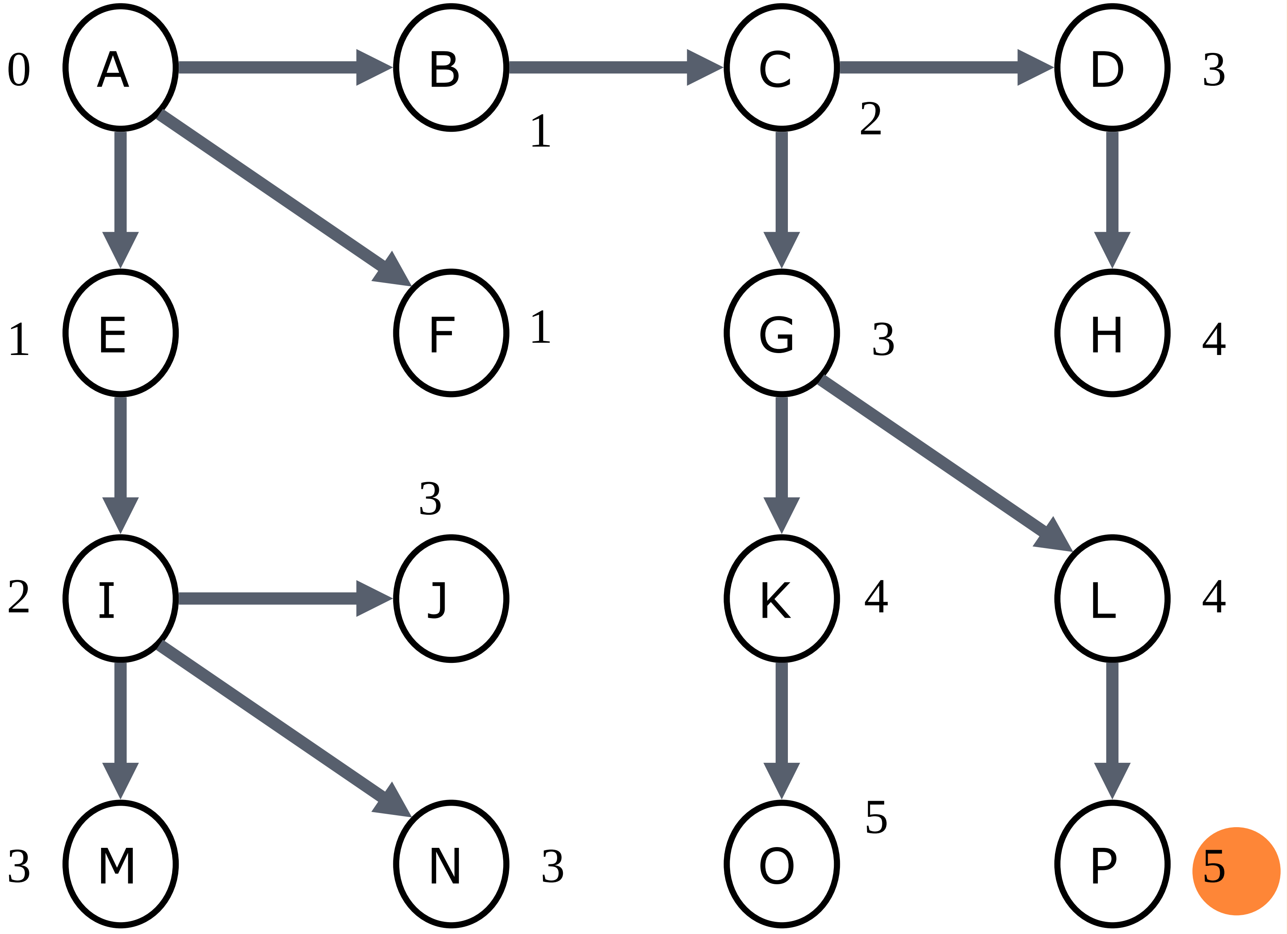

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
0
1
2
1
1
2


0
A
B
1
C
2
D
3
1
E
F
1
G
3
H
3
2
I
J
K
L
3
M
N
3
O
P


0
A
B
C
D
3
1
2
E
F
1
G
3
H
4
1
3
I
J
K
4
L
4
2
M
N
O
P
3
3


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
0
1
2
3
1
1
3
4
3
2
4
4
3
3
5
5


A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
0
1
2
3
1
1
3
4
2
3
4
4
3
3
5
5


0
A
B
1
C
2
D
3
E
1
F
1
G
3
H
4
3
I
2
J
K
4
L
4
M
3
N
3
O
5
P
5

Thnk u…very much..