
# CSE408
# BFS, DFS, Connected components

Lecture #14

# Graph Traversal


**Topics**

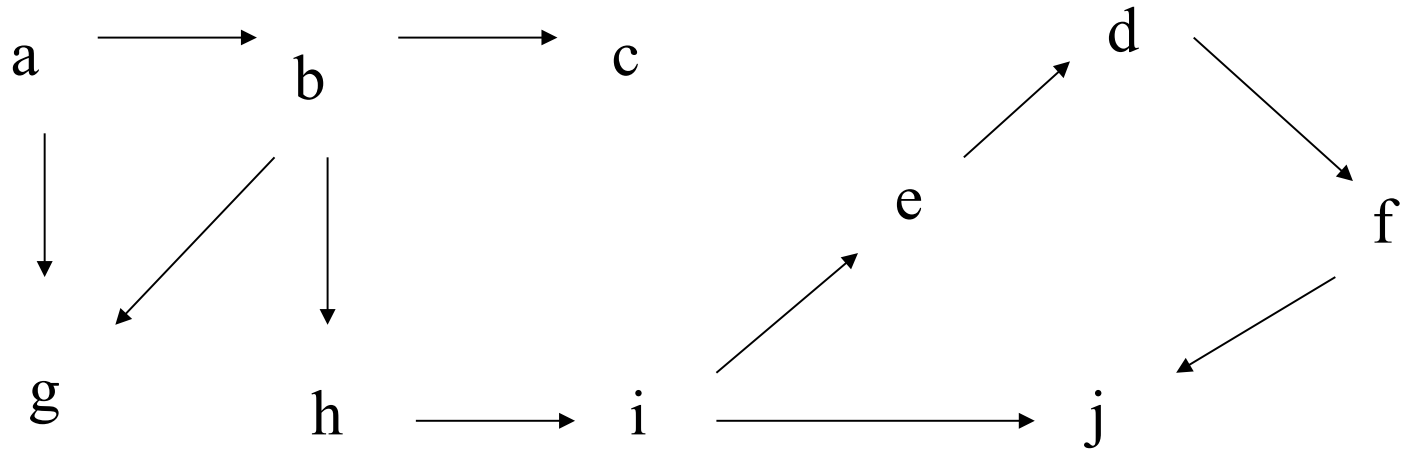
- Depth First Search (DFS)
- Breadth First Search (BFS)

# Graph Search (traversal)


- How do we search a graph?
  - At a particular vertices, where shall we go next?
- Two common framework:
    - the depth-first search (DFS)
    - the breadth-first search (BFS) and
  - In DFS, go as far as possible along a single path until reach a dead end (a vertex with no edge out or no neighbor unexplored) then backtrack
  - In BFS, one explore a graph level by level away (explore all neighbors first and then move on)

# Depth-First Search (DFS)

- The basic idea behind this algorithm is that it traverses the graph using recursion
  - Go as far as possible until you reach a deadend
  - Backtrack to the previous path and try the next branch
  - The graph below, started at node a, would be visited in the following order: a, b, c, g, h, i, e, d, f, j



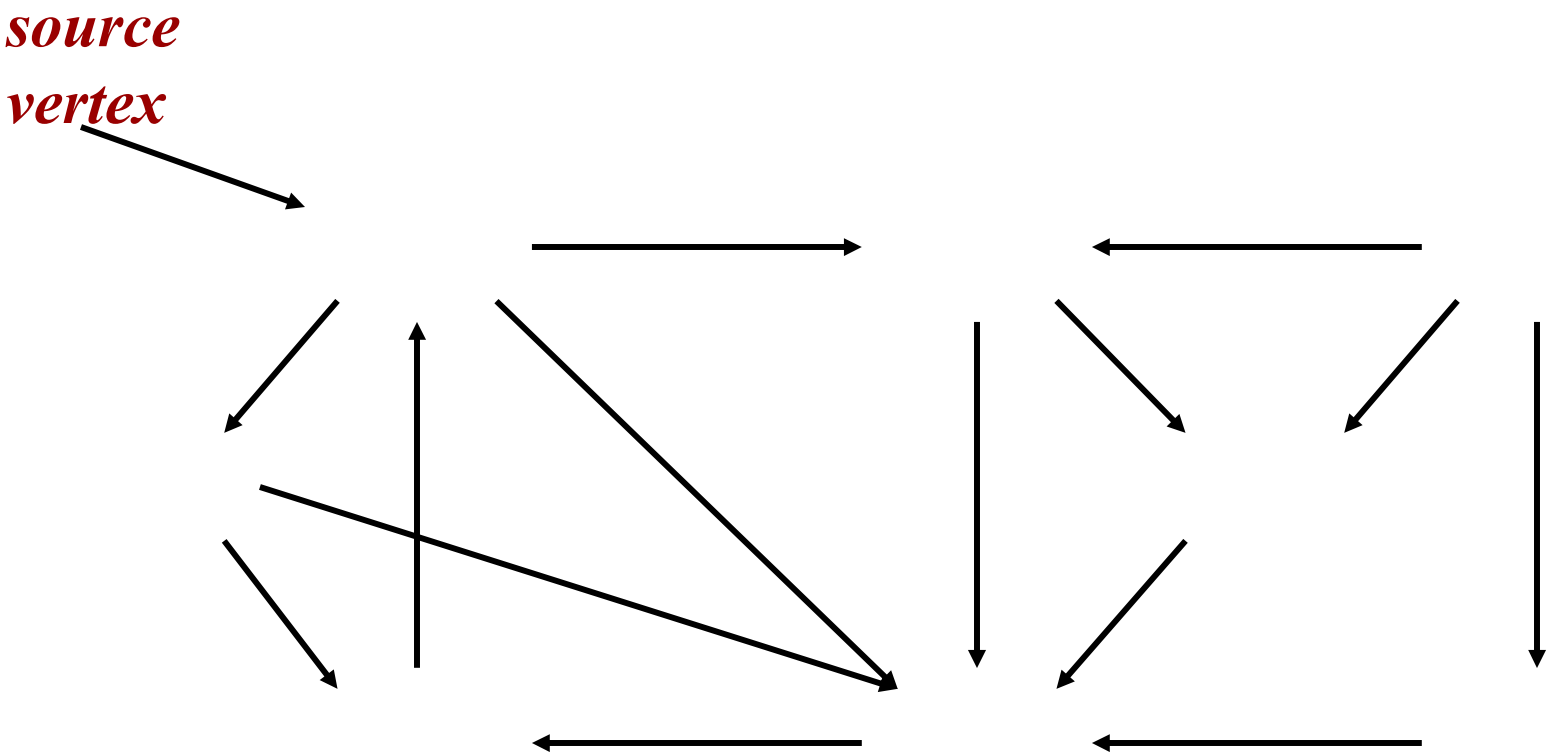
# DFS: Color Scheme


- Vertices initially colored white
- Then colored gray when discovered
- Then black when finished

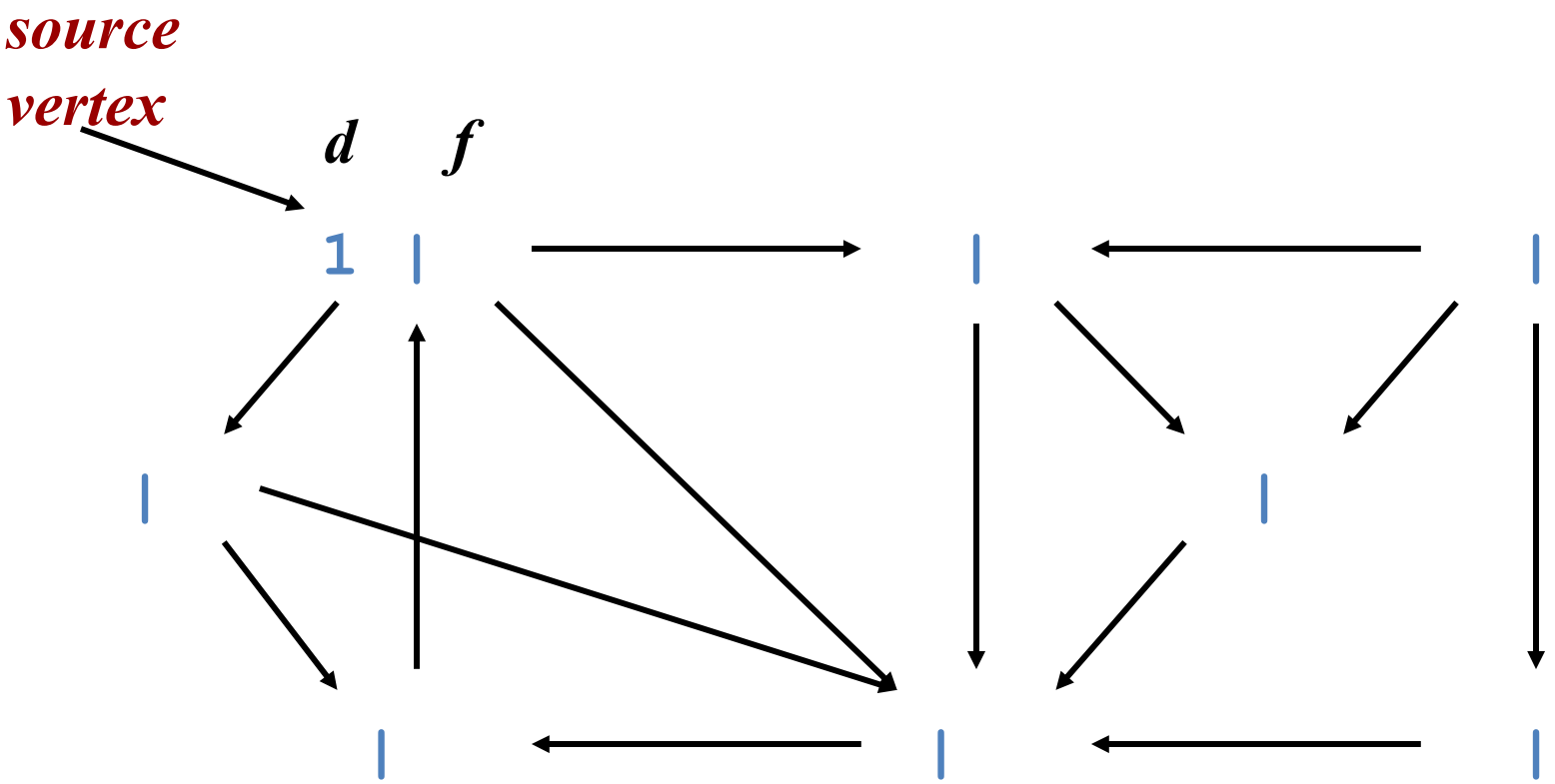
# DFS: Time Stamps


- Discover time d[u]:  when u is first discovered
- Finish time f[u]: when backtrack from u
- d[u] < f[u]

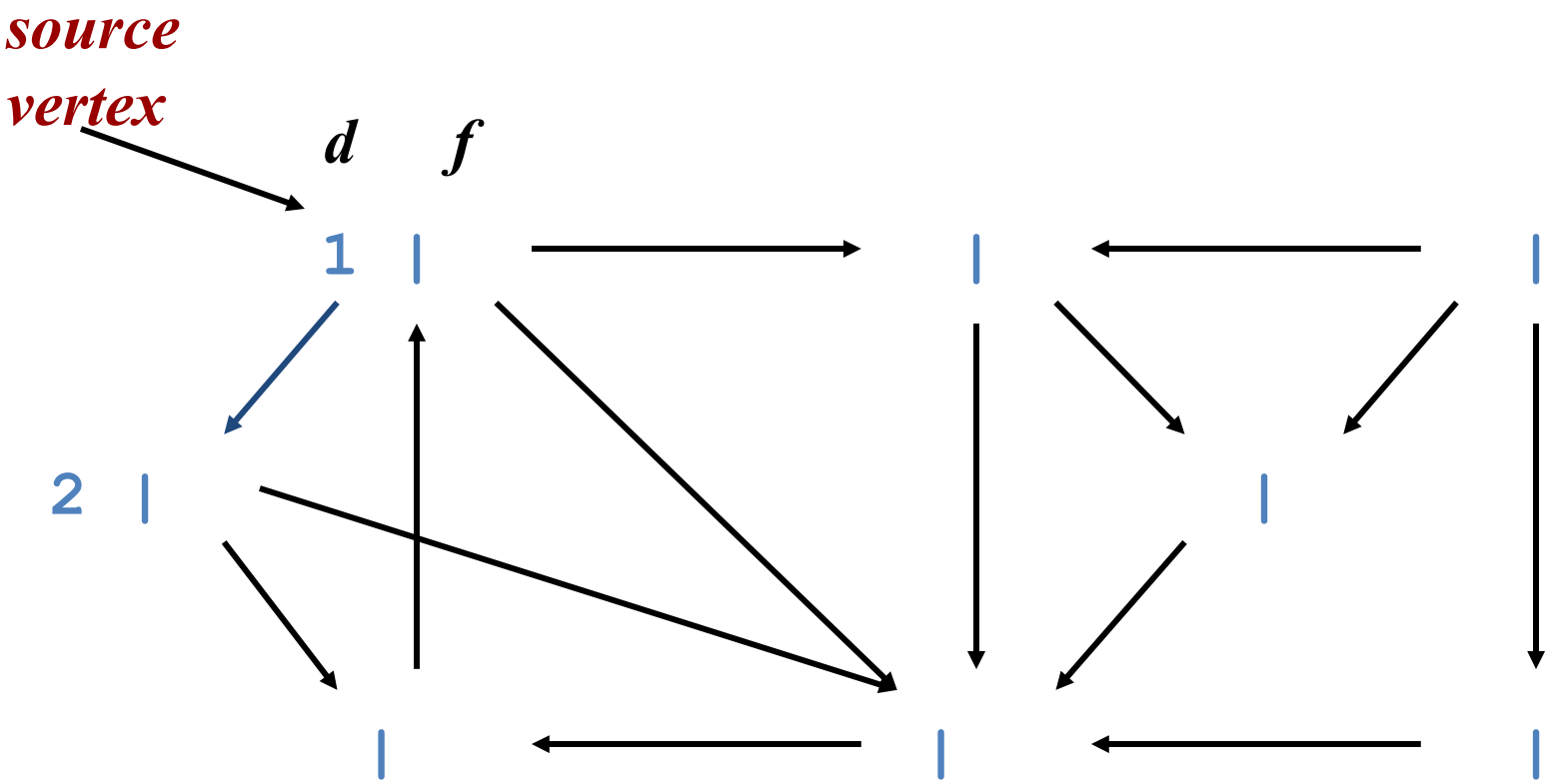
# DFS Example



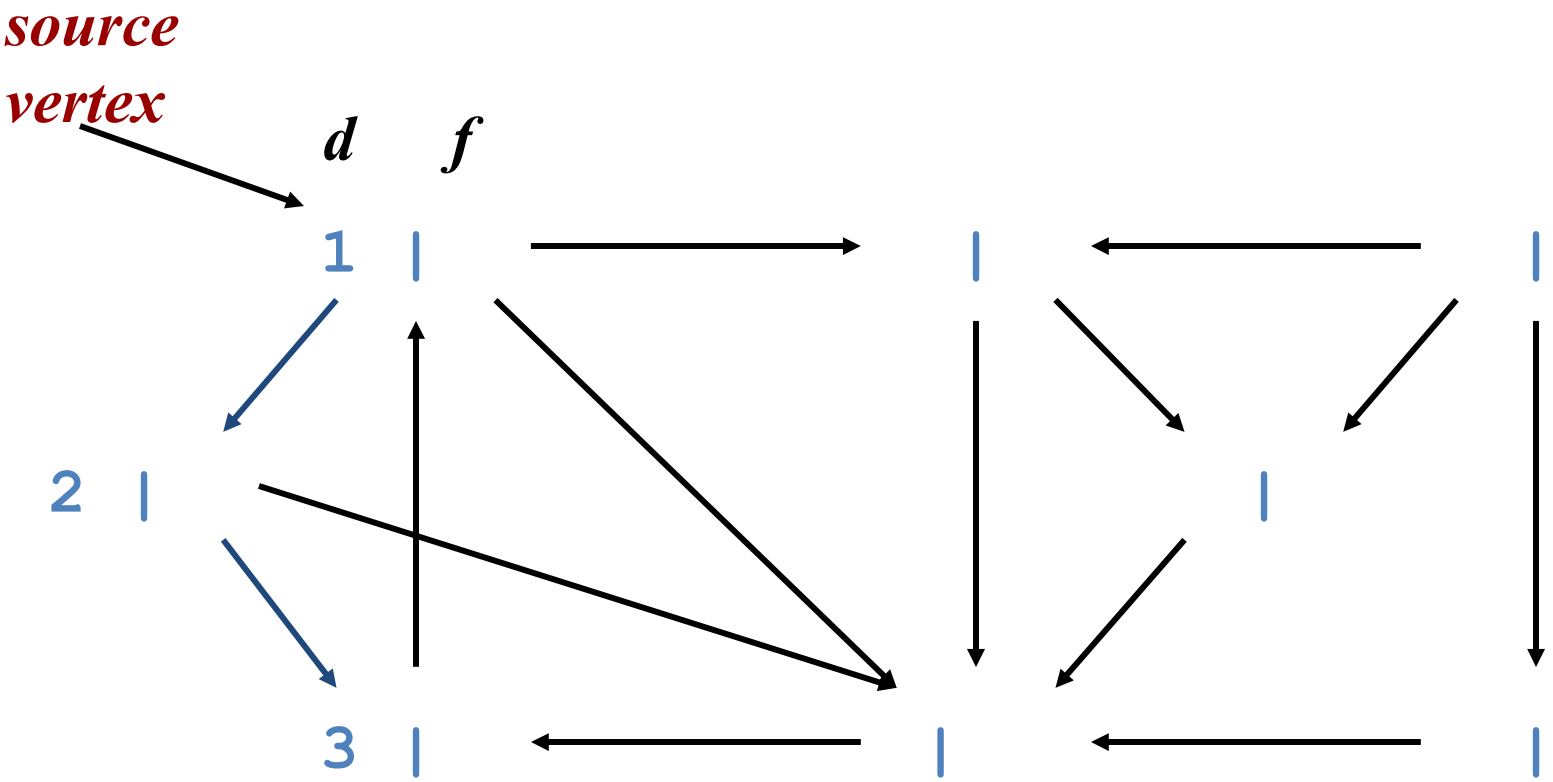
# DFS Example



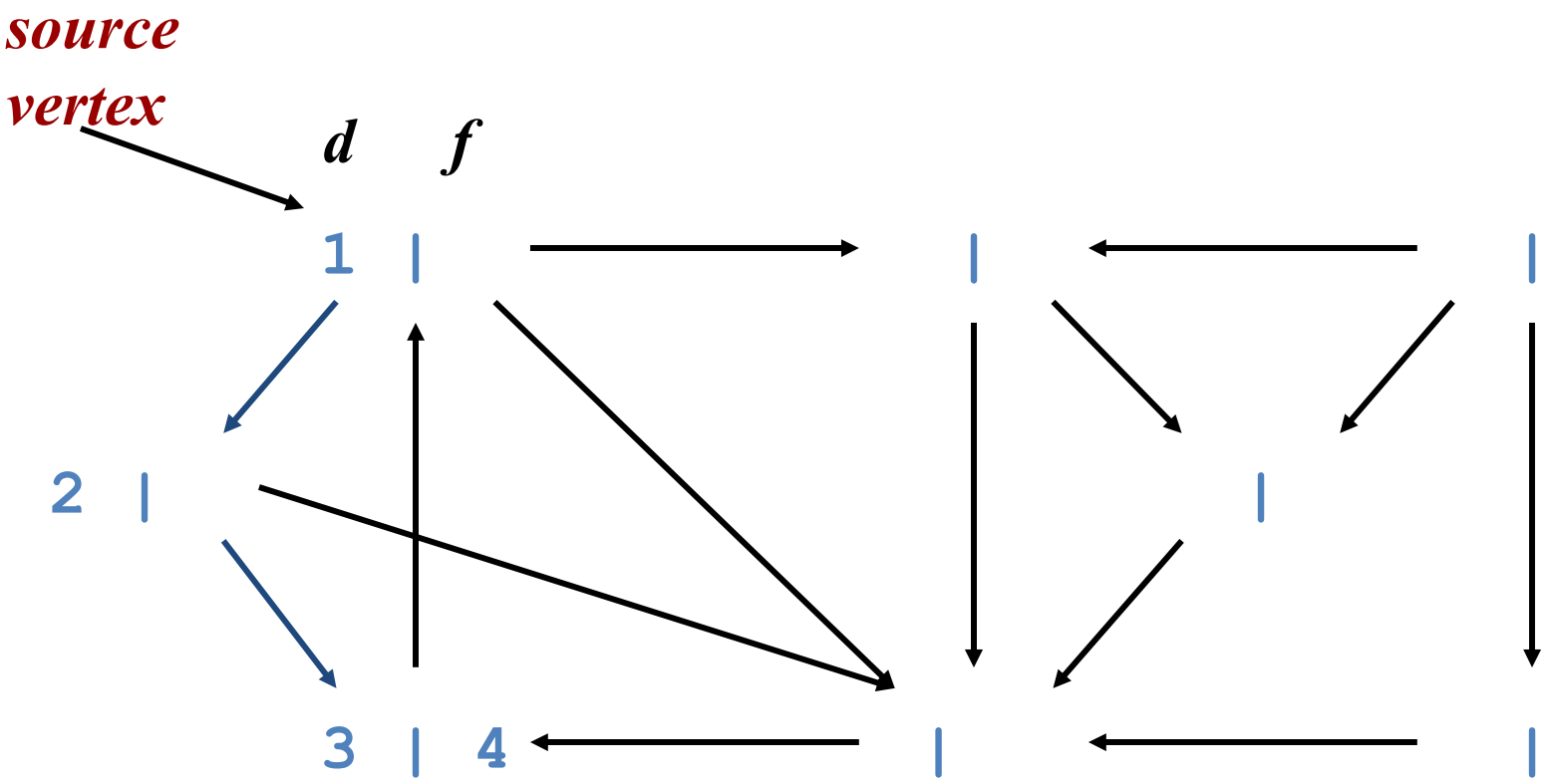
# DFS Example



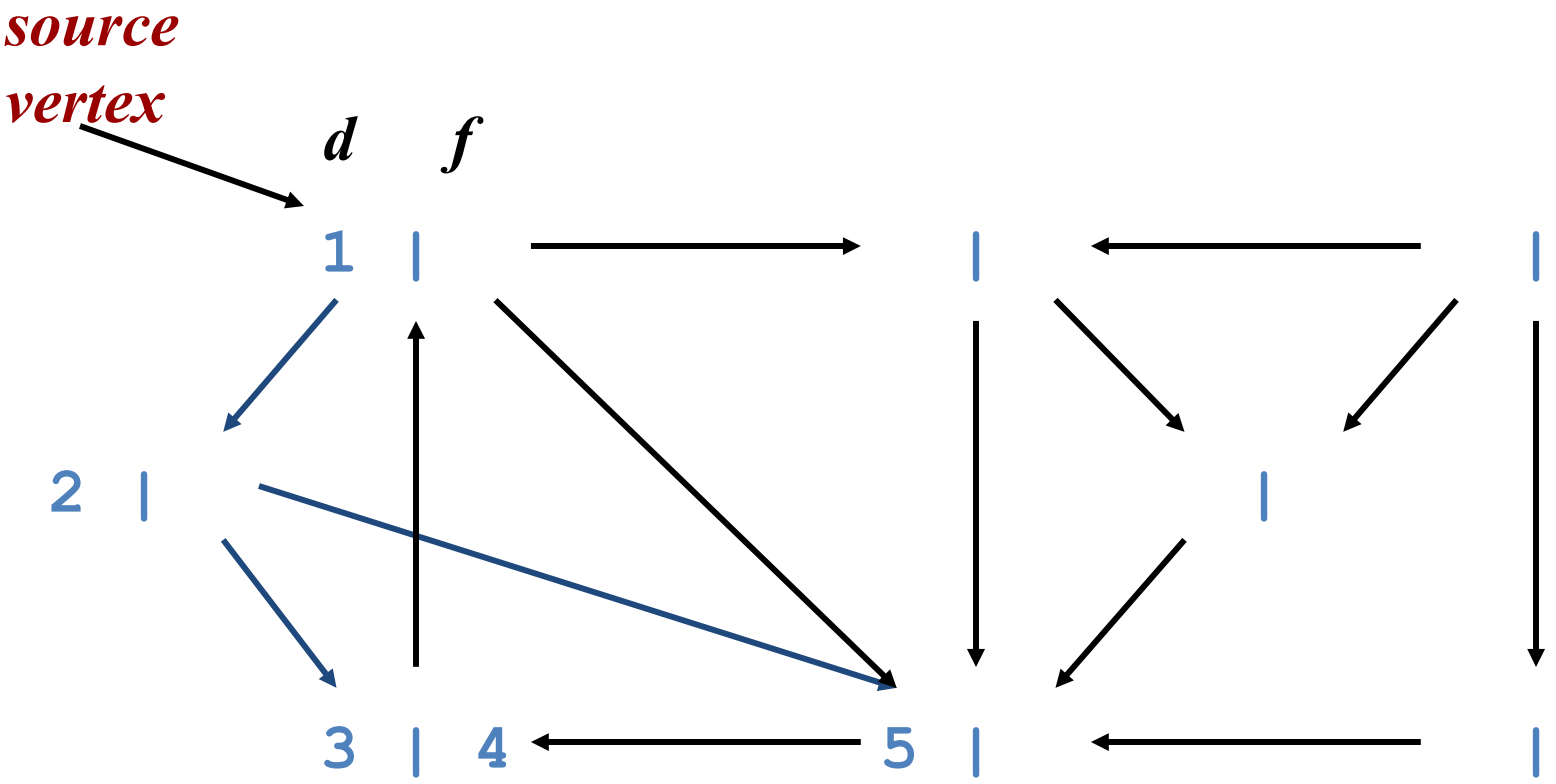
# DFS Example



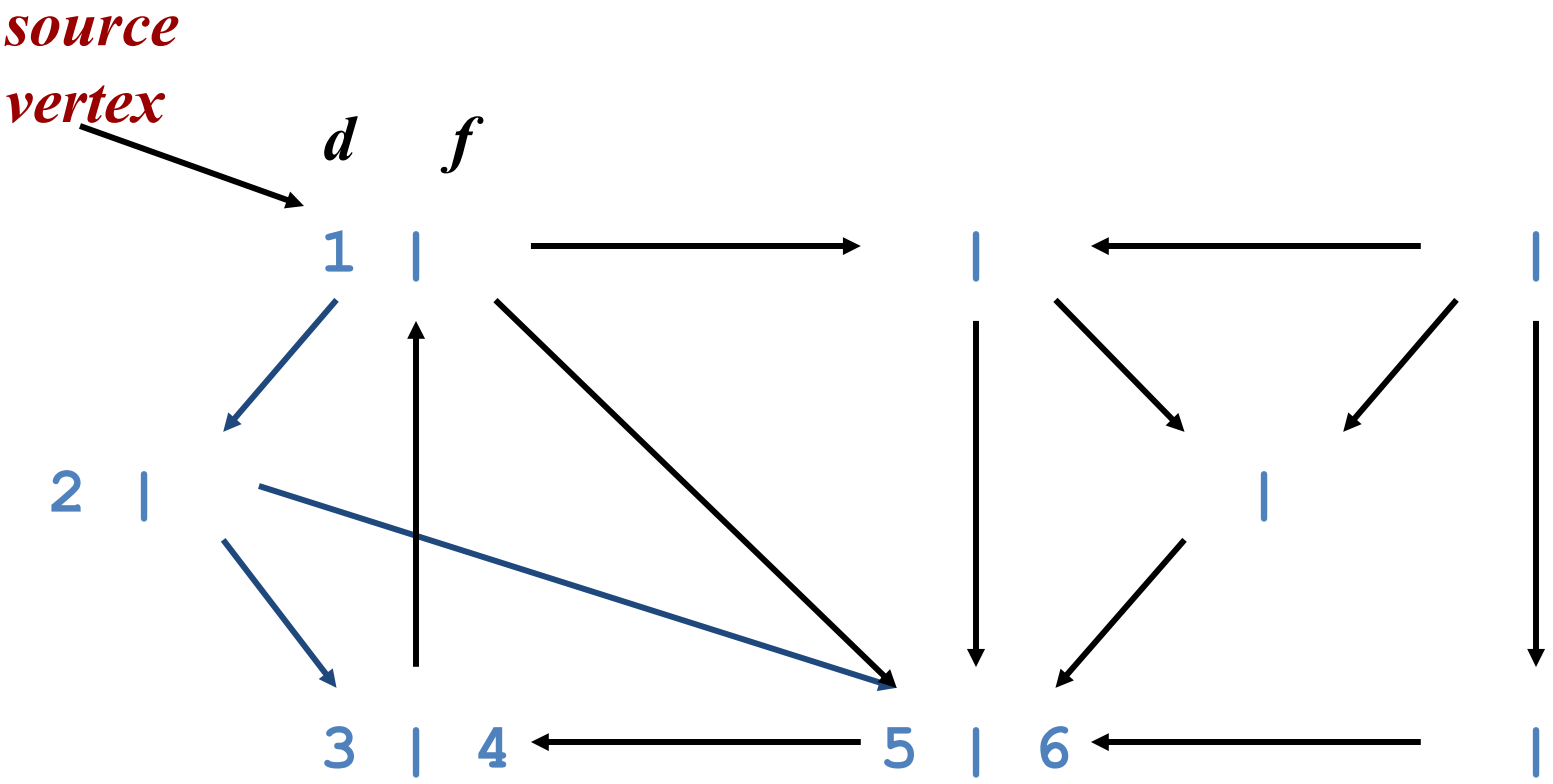
# DFS Example



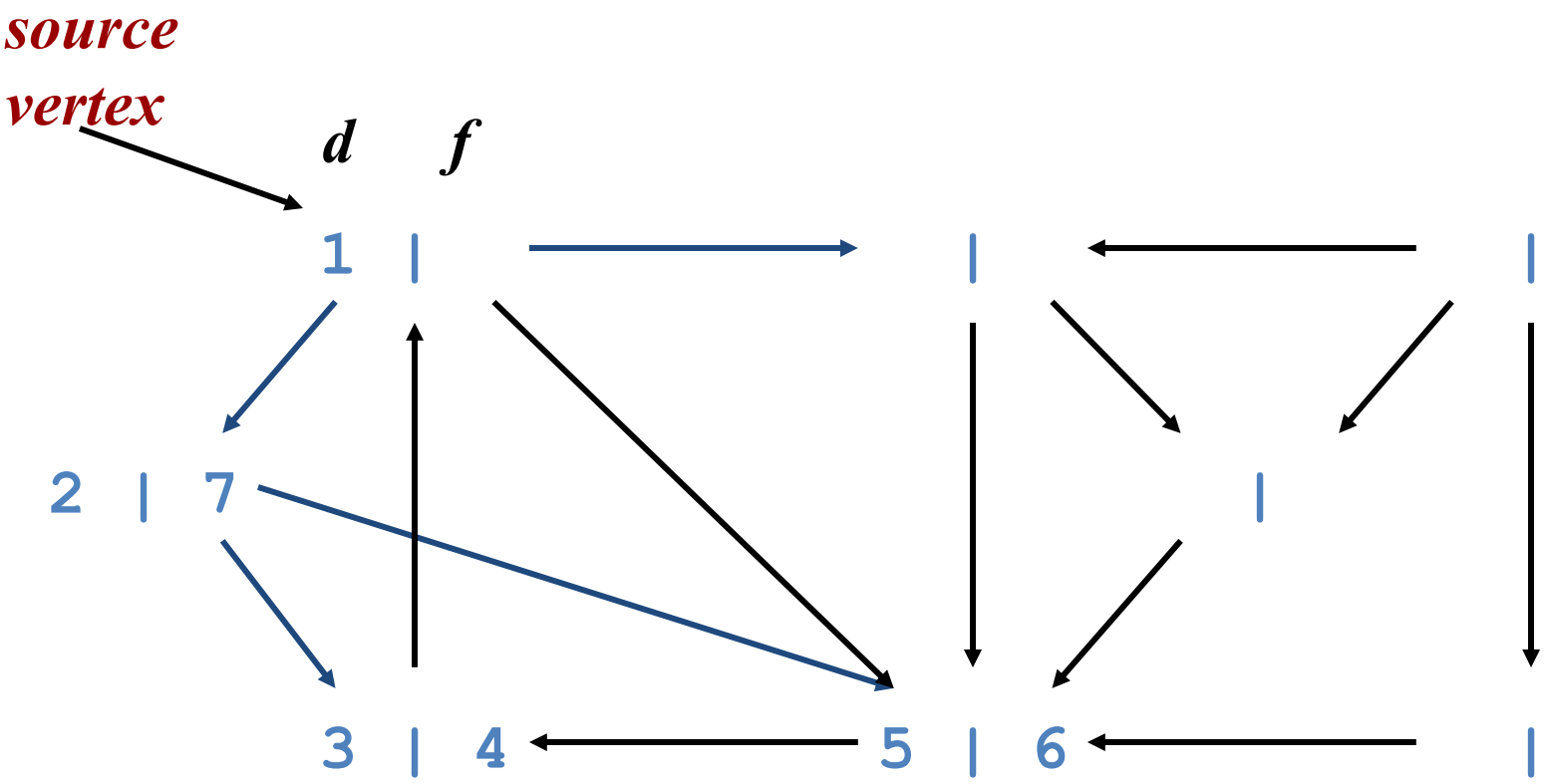
# DFS Example



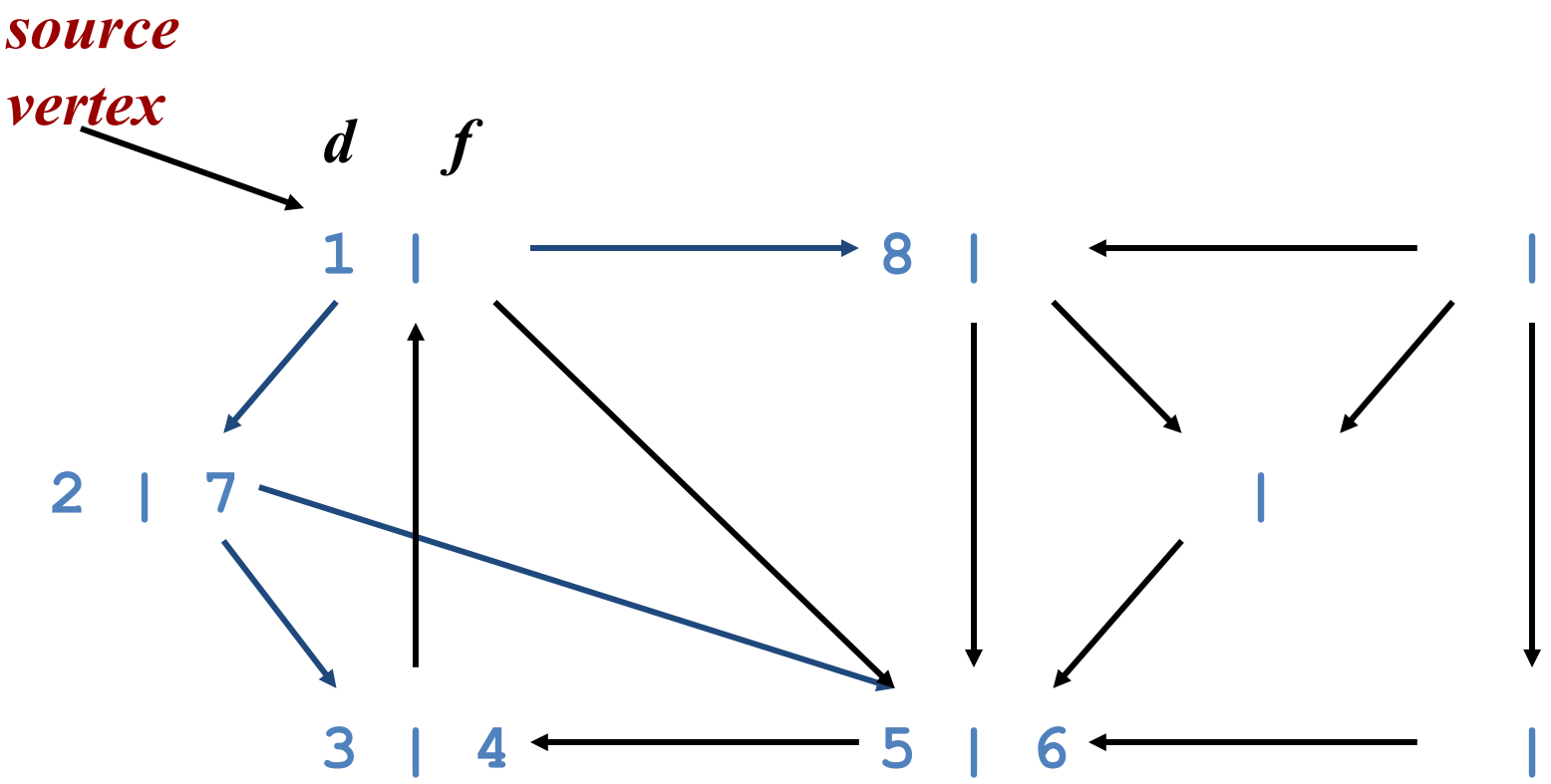
# DFS Example



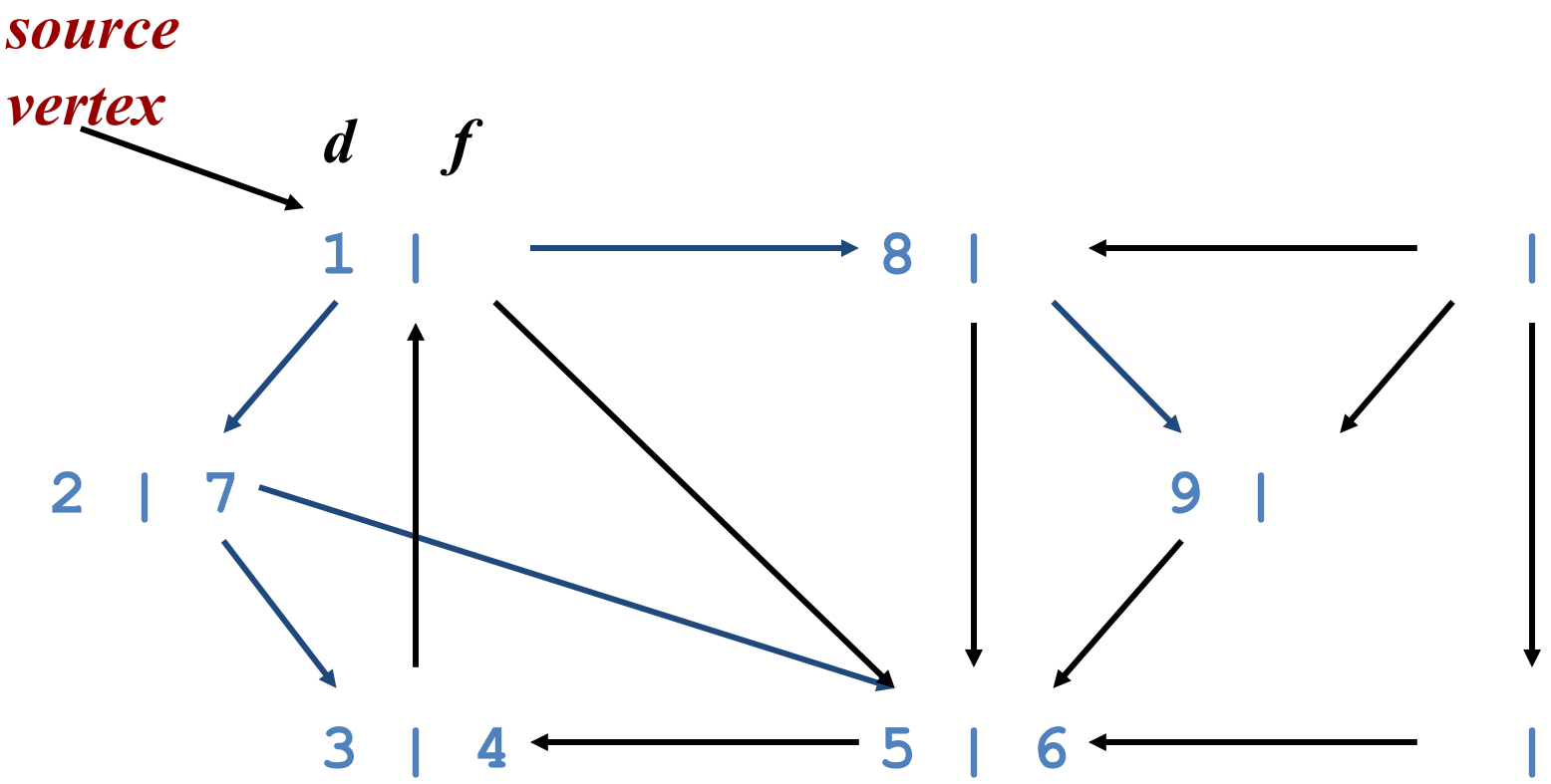
# DFS Example



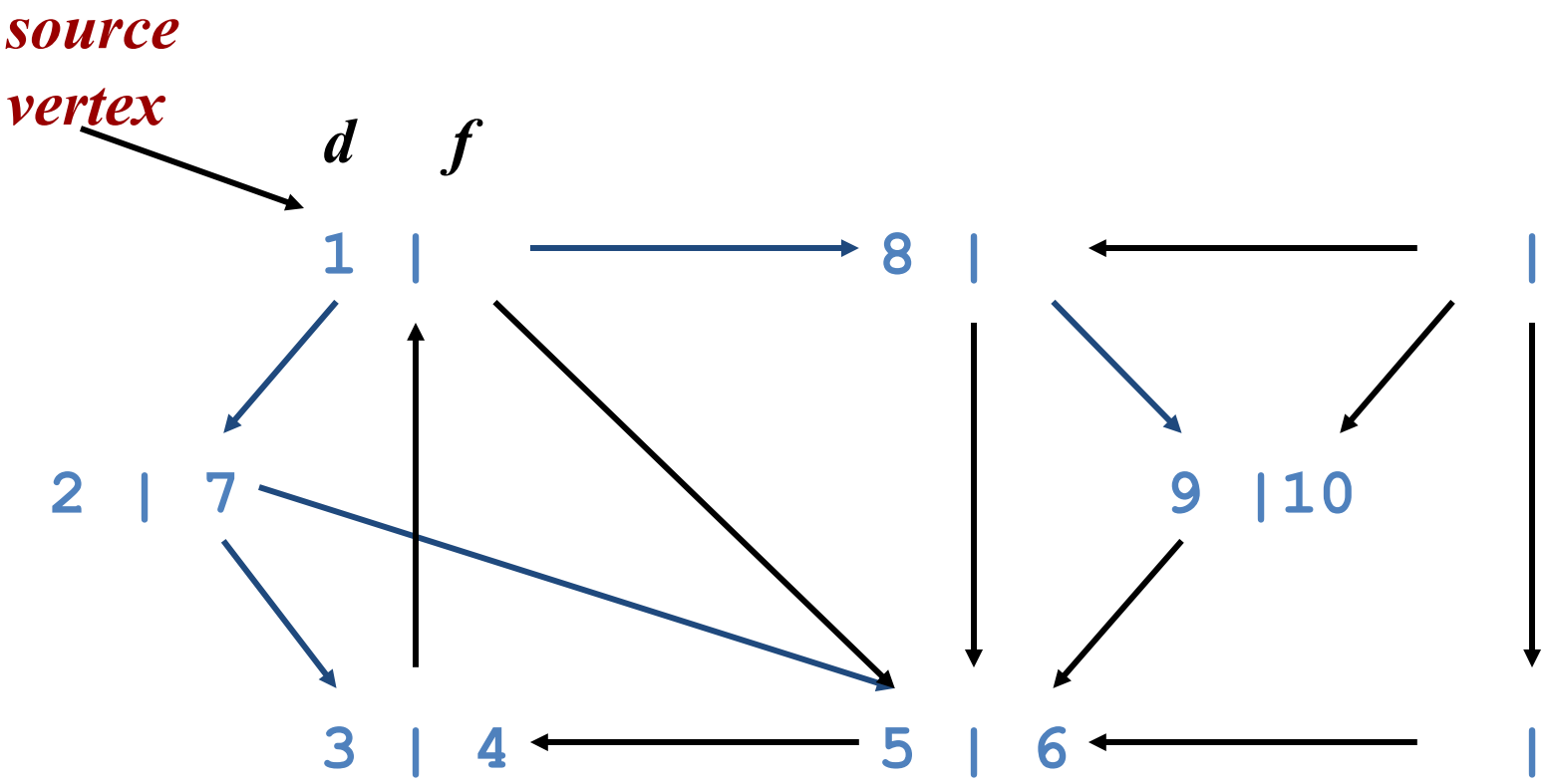
# DFS Example



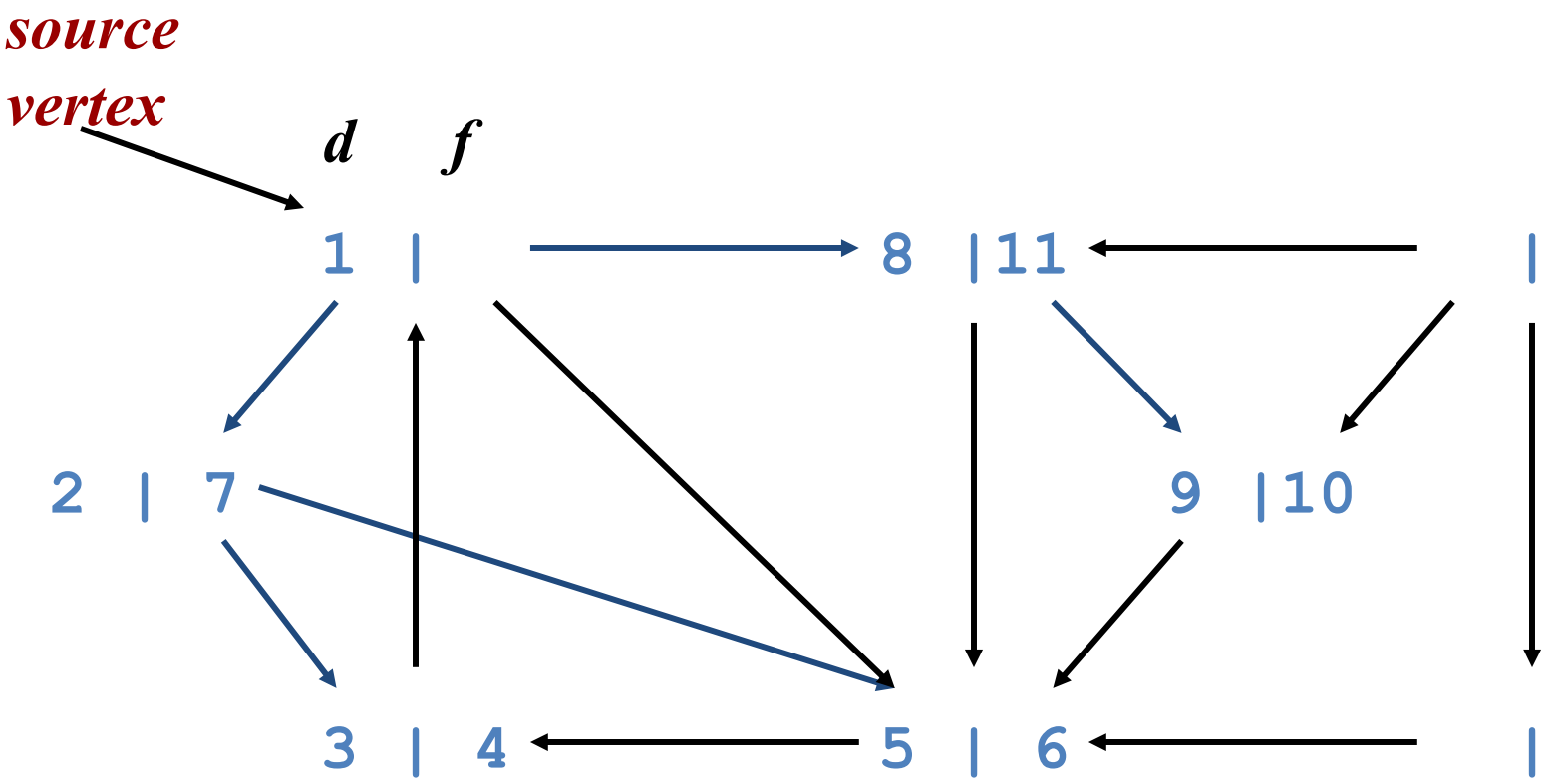
# DFS Example



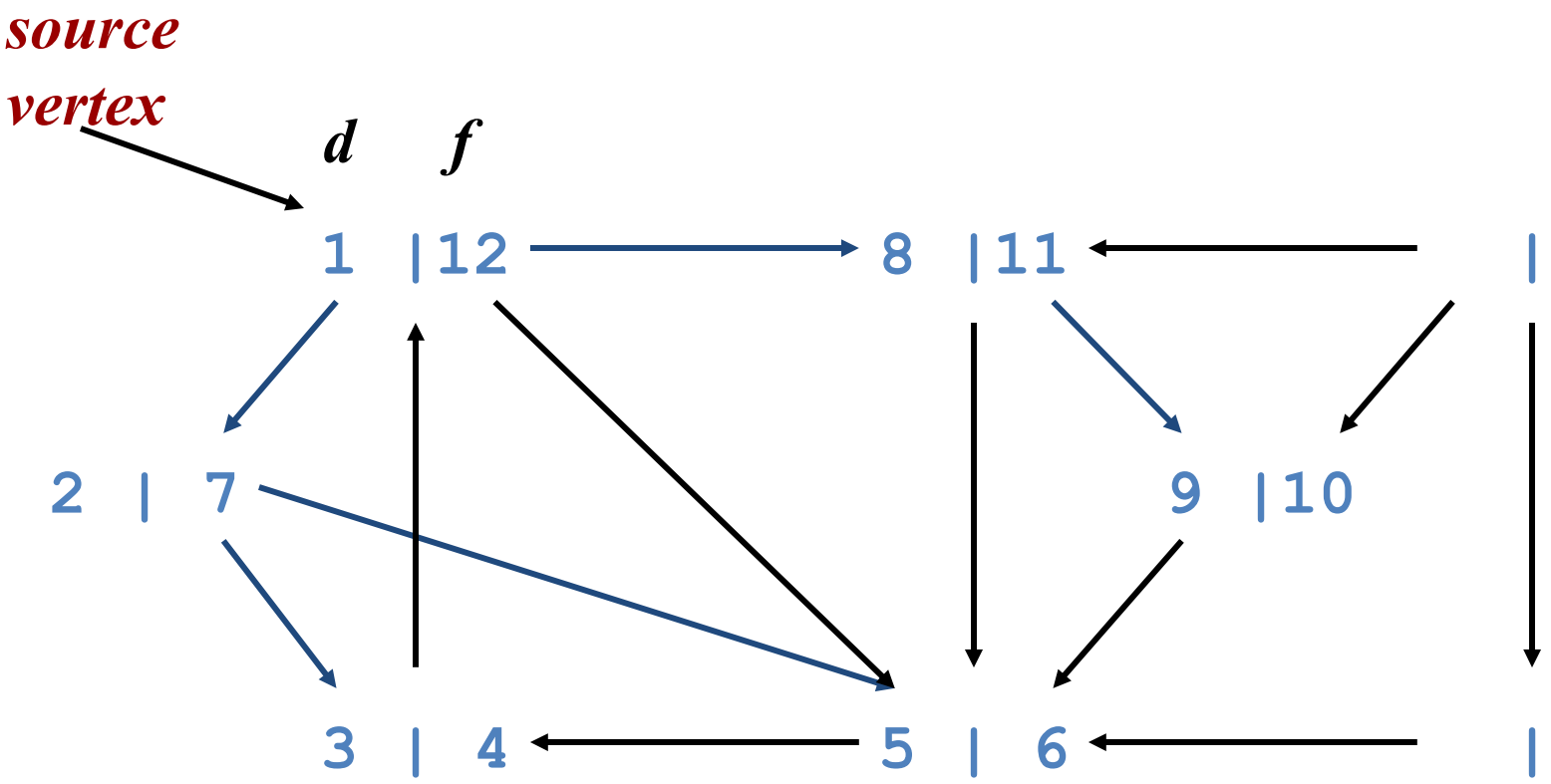
# DFS Example



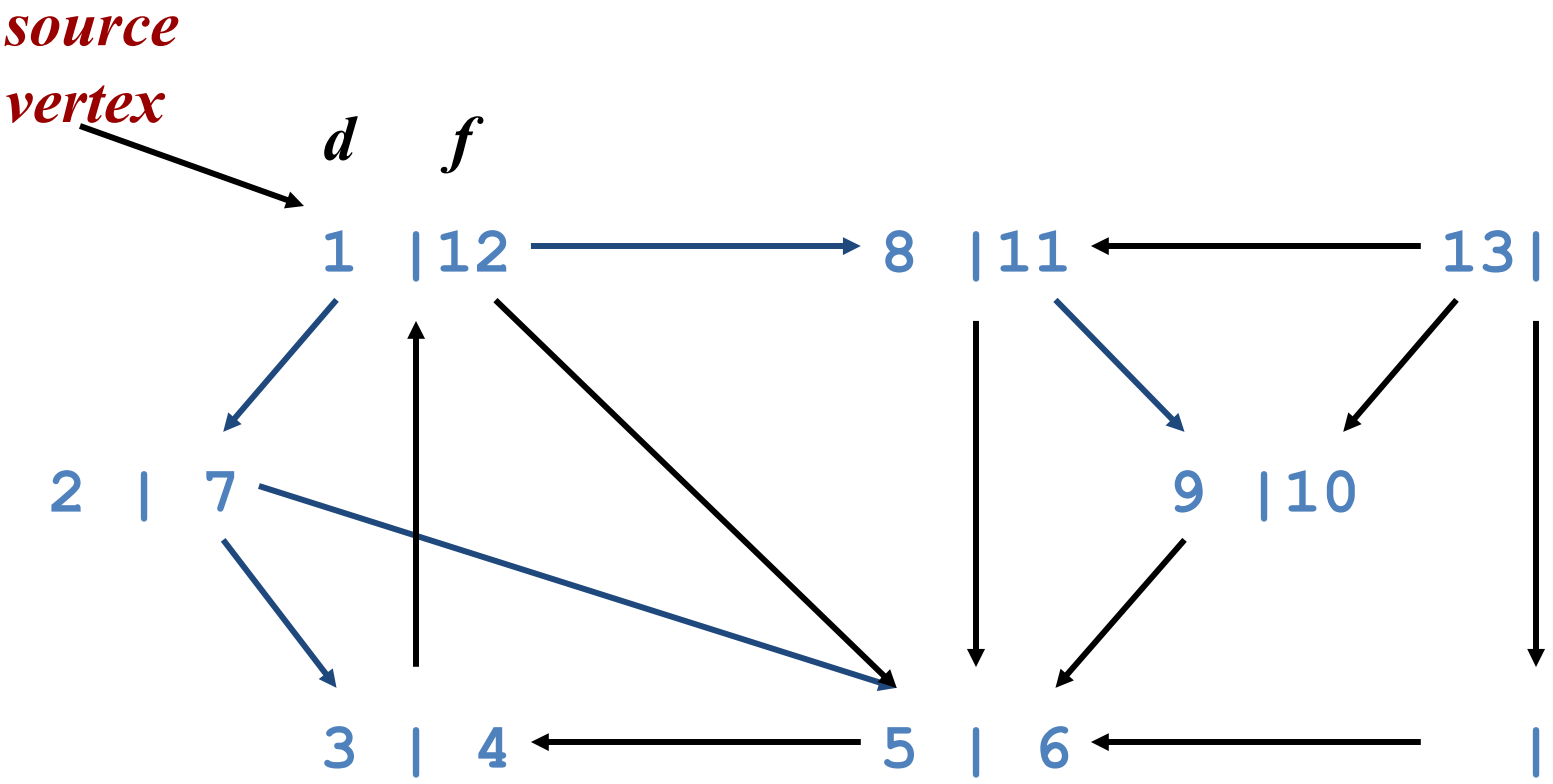
# DFS Example



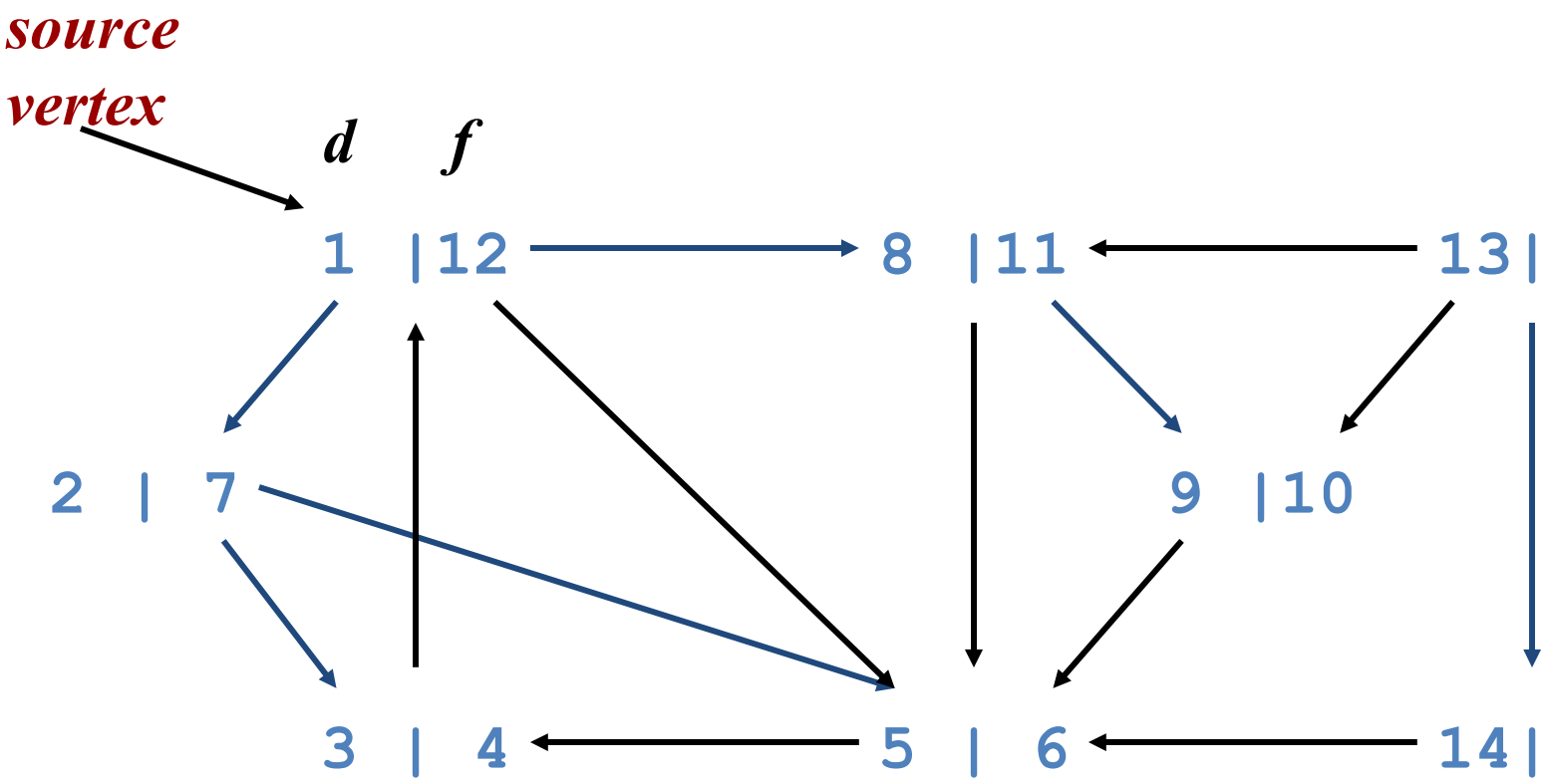
# DFS Example



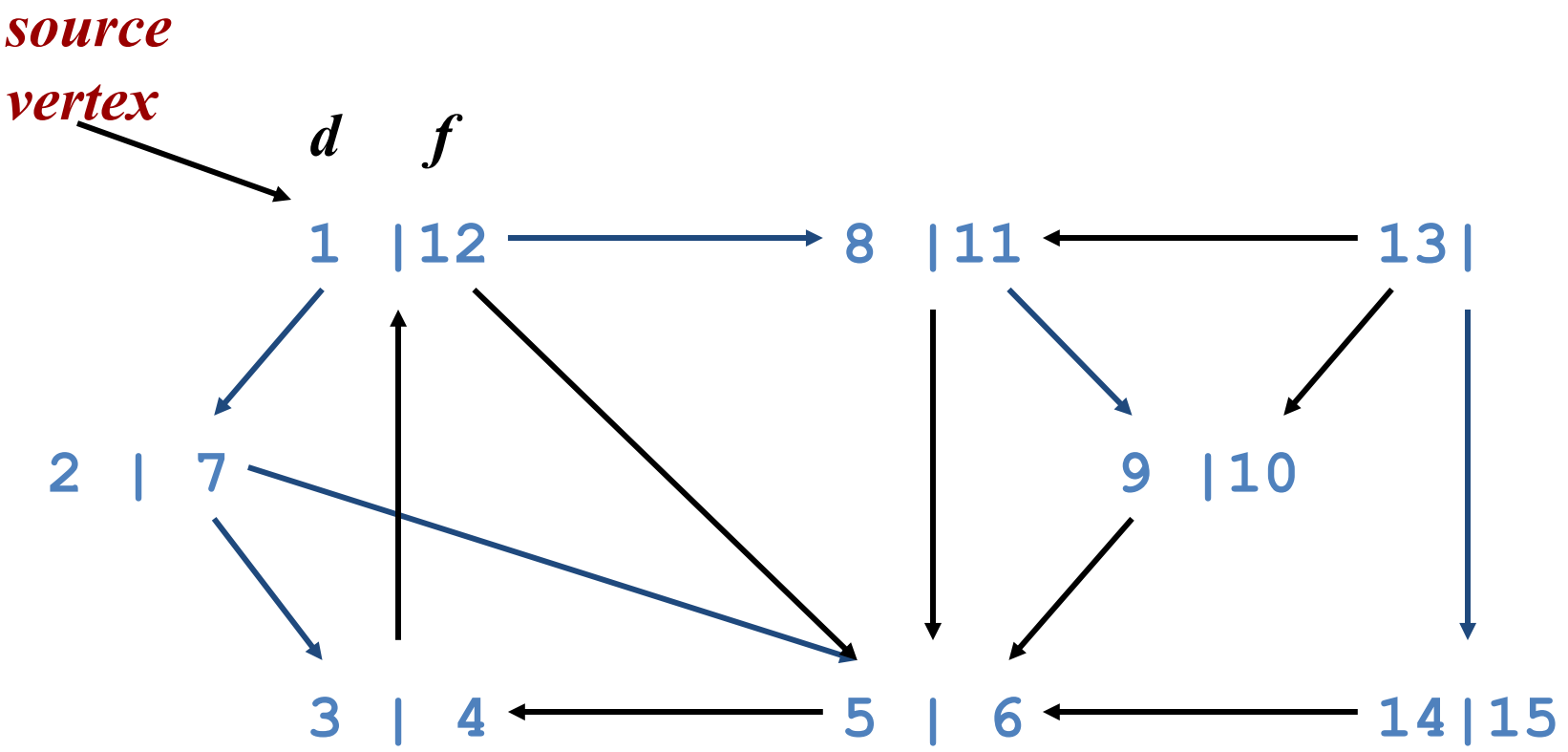
# DFS Example



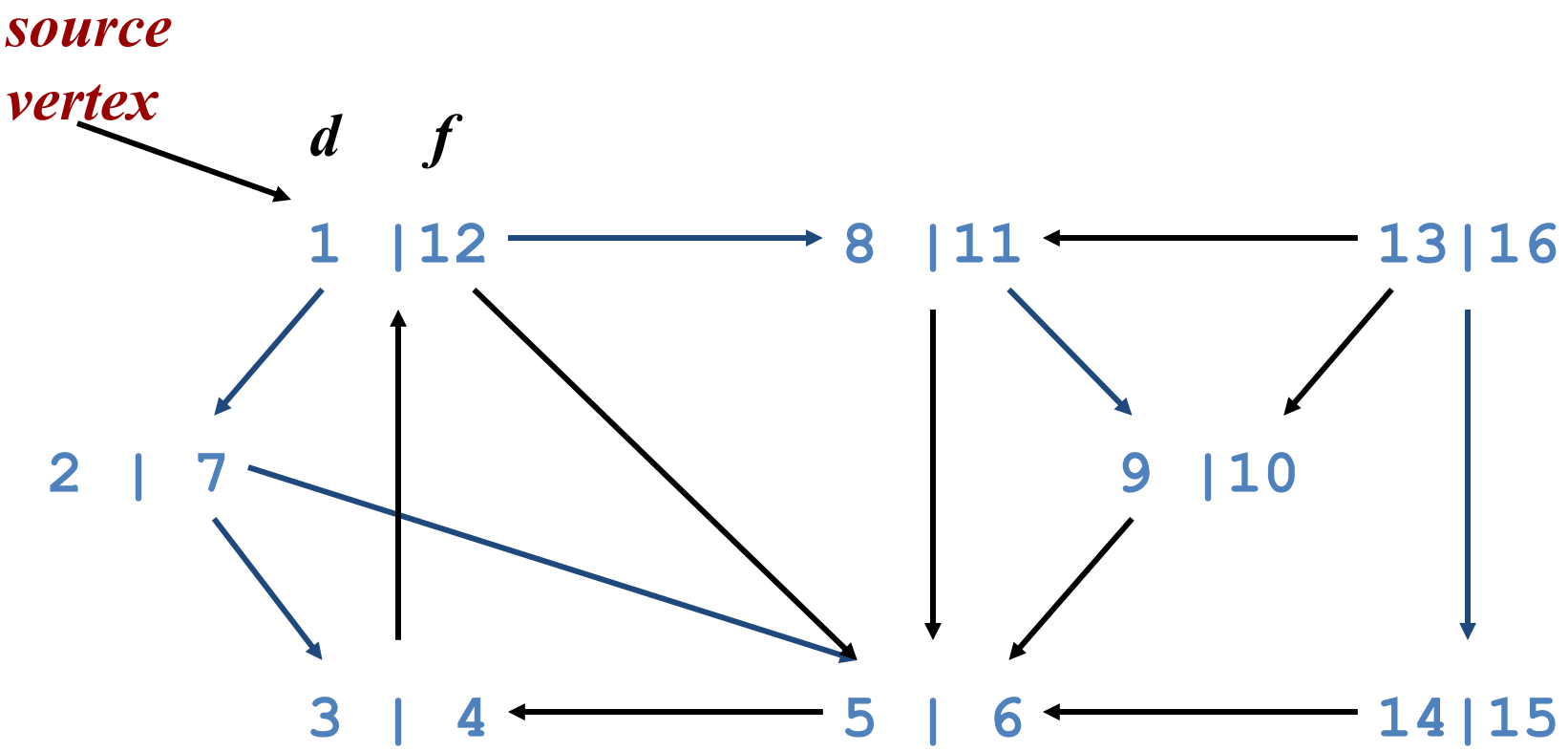
# DFS Example



# DFS Example



# DFS Example



# DFS: Algorithm

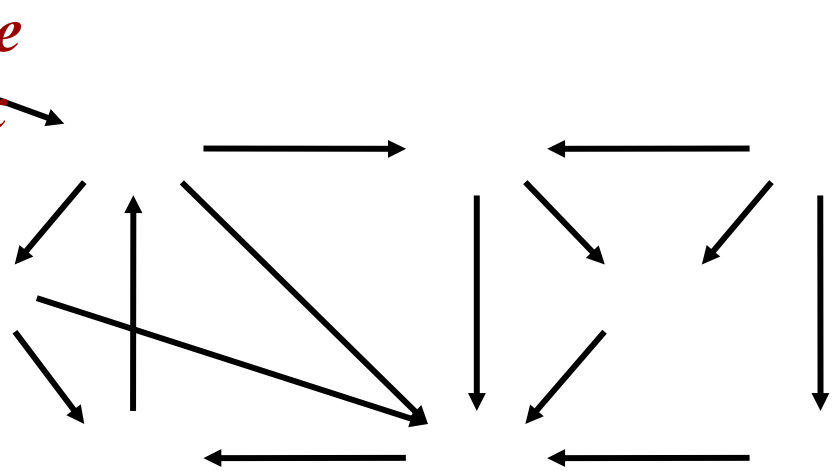
DFS(G)

- for each vertex u in V,
- color[u]=white; π[u]=NIL
- time=0;
- for each vertex u in V
  - if (color[u]=white)
    - DFS-VISIT(u)

# DFS: Algorithm (Cont.)

DFS-VISIT(u)

- color[u]=gray;
- time = time + 1;
- d[u] = time;
- for each v in Adj(u) do
- if (color[v] = white)
- $\pi$[v] = u;
- DFS-VISIT(v);
- color[u] = black;
- time = time + 1; f[u]= time;



# DFS: Complexity Analysis


Initialization complexity is O(V)

DFS_VISIT is called exactly once for each vertex

And DFS_VISIT scans all the edges which causes cost of O(E)

Overall complexity is O(V + E)

# DFS: Application


- Topological Sort
- Strongly Connected Component

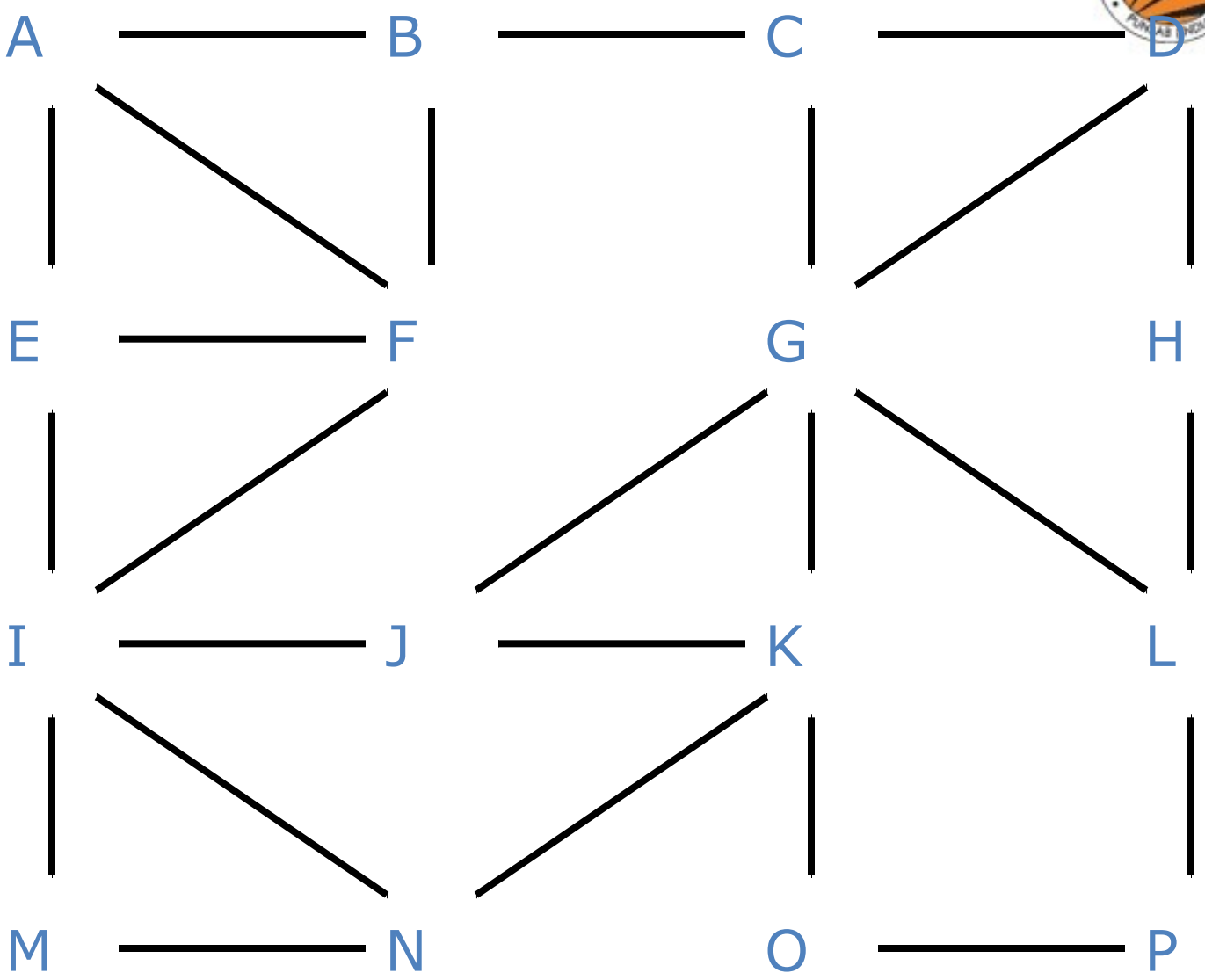
# Breadth-first Search (BFS)

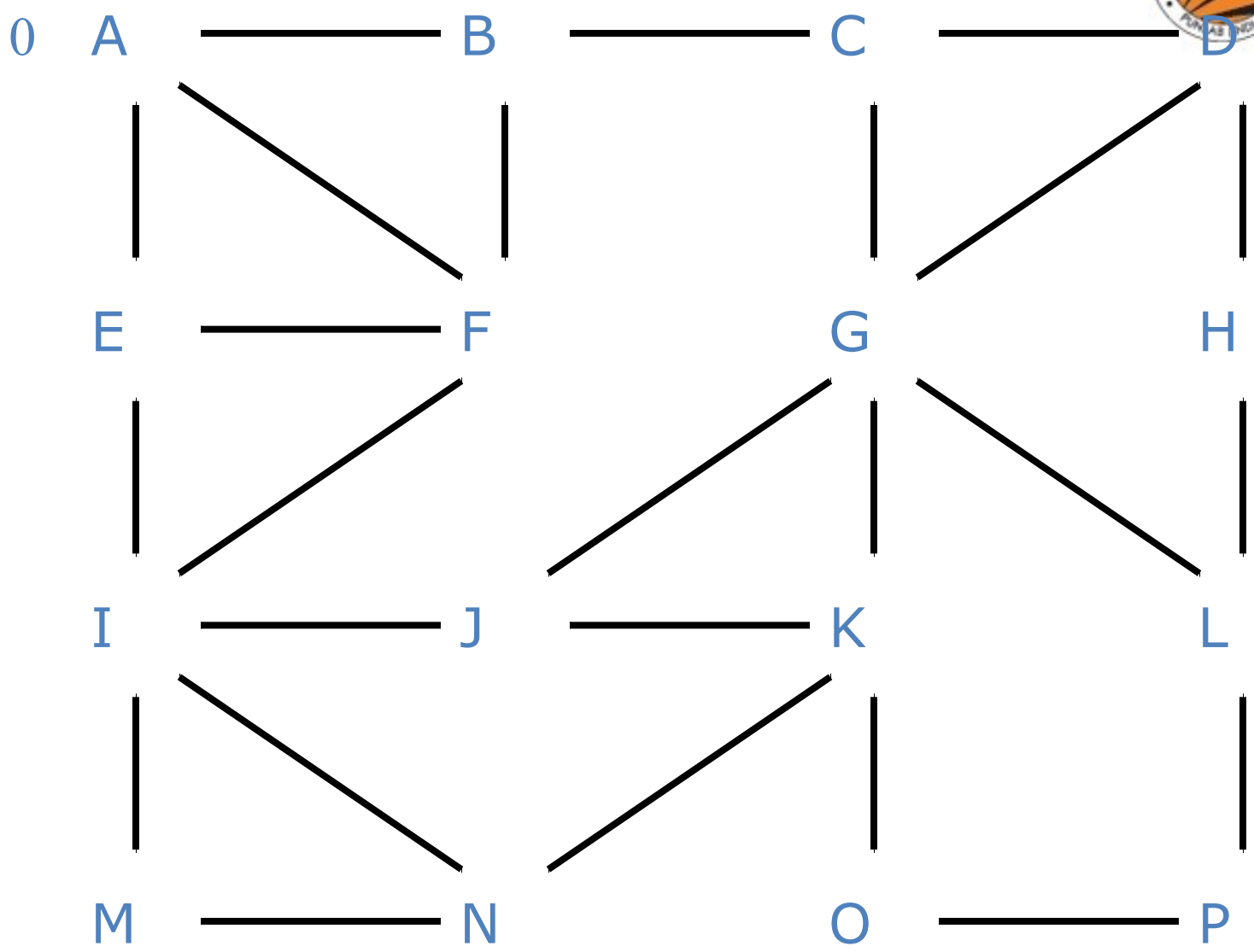

- Search for all vertices  that are directly reachable from the root (called level 1 vertices)
- After mark all these vertices, visit all vertices that are directly reachable from any level 1 vertices (called level 2 vertices), and so on.
- In general, level k vertices are directly reachable from a level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

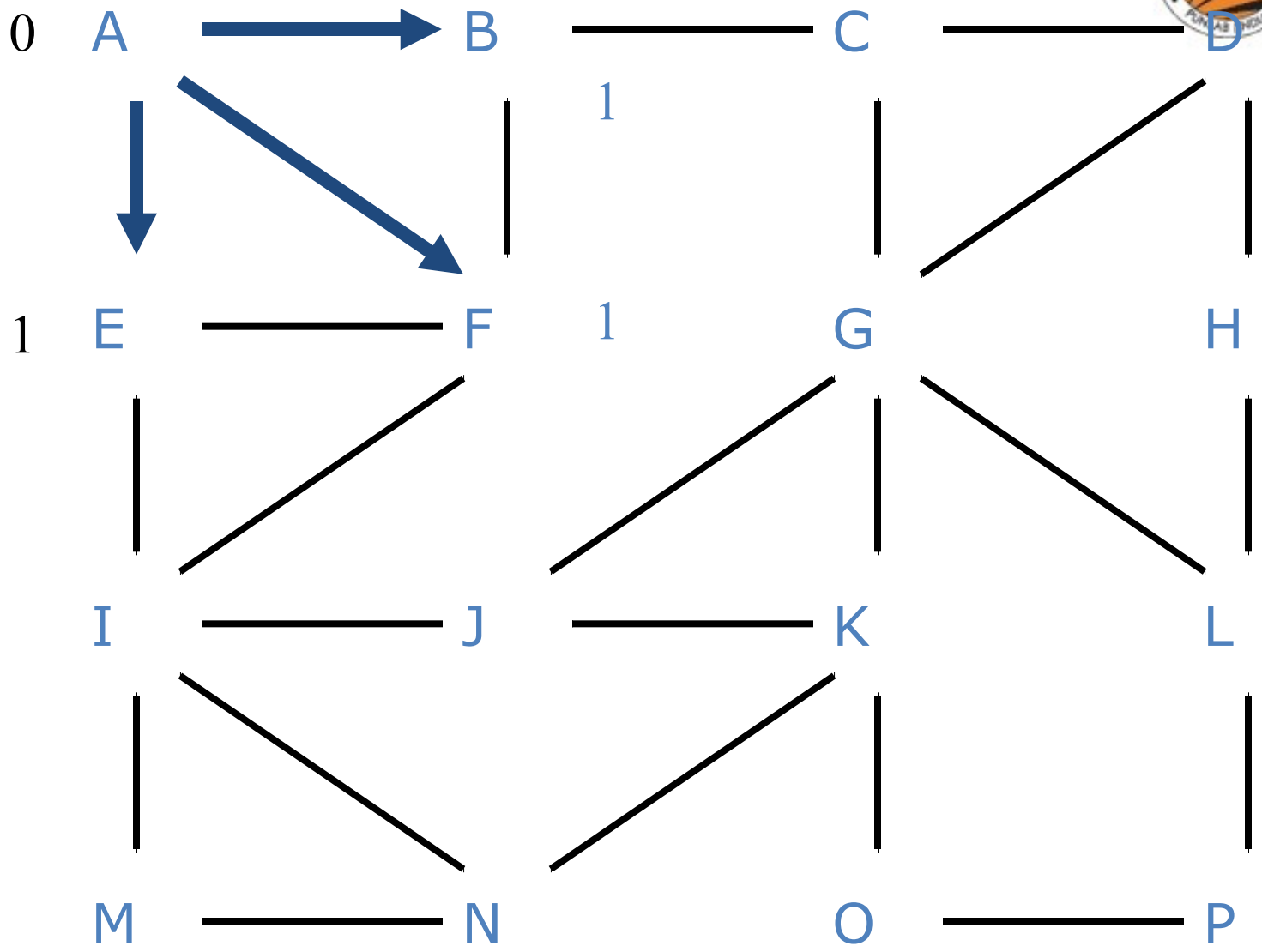
# BFS: the Color Scheme

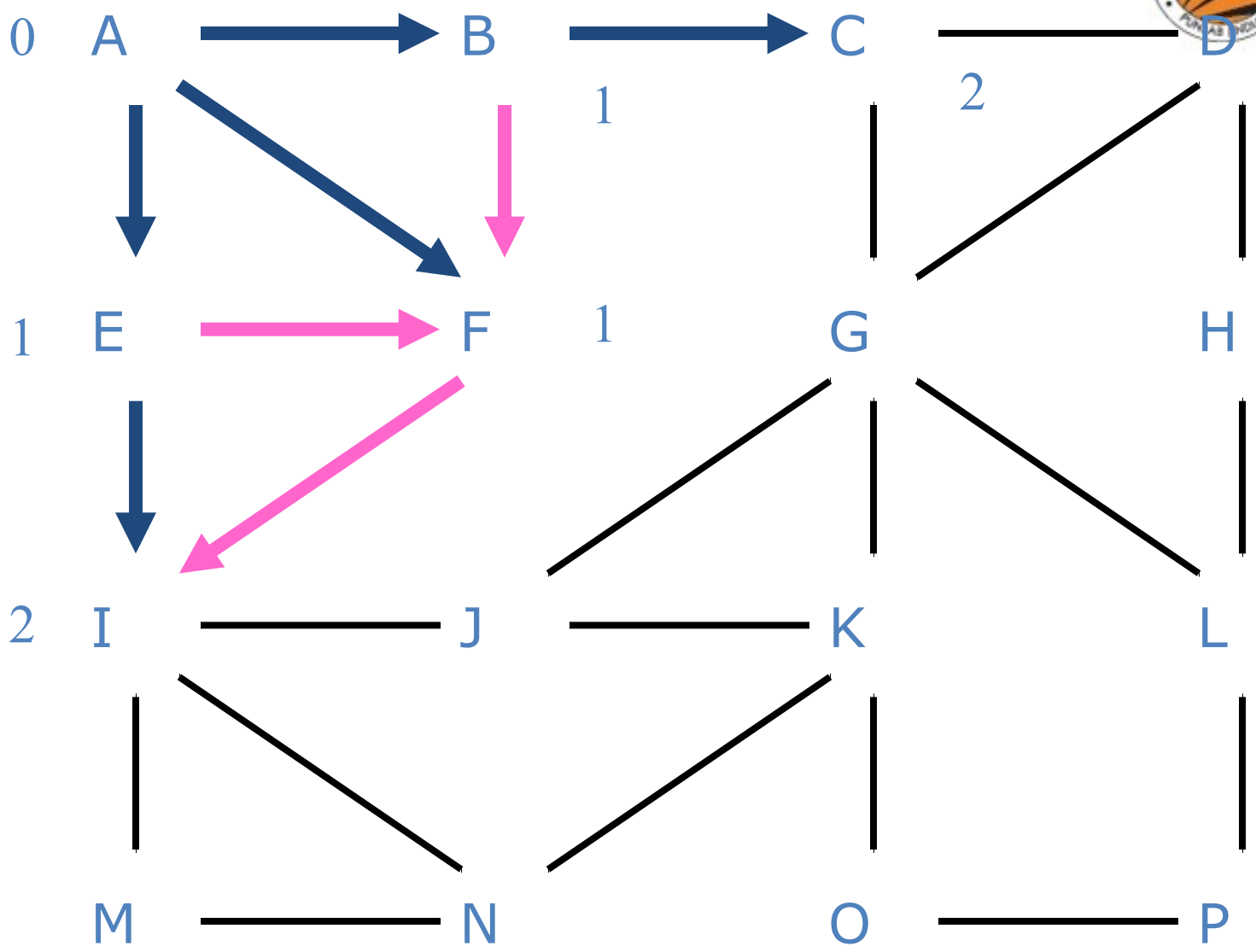

- White vertices have not been discovered
  - All vertices start out white
- Grey vertices are discovered but not fully explored
  - They may be adjacent to white vertices
- Black vertices are discovered and fully explored
  - They are adjacent only to black and gray vertices
- Explore vertices by scanning adjacency list of grey vertices

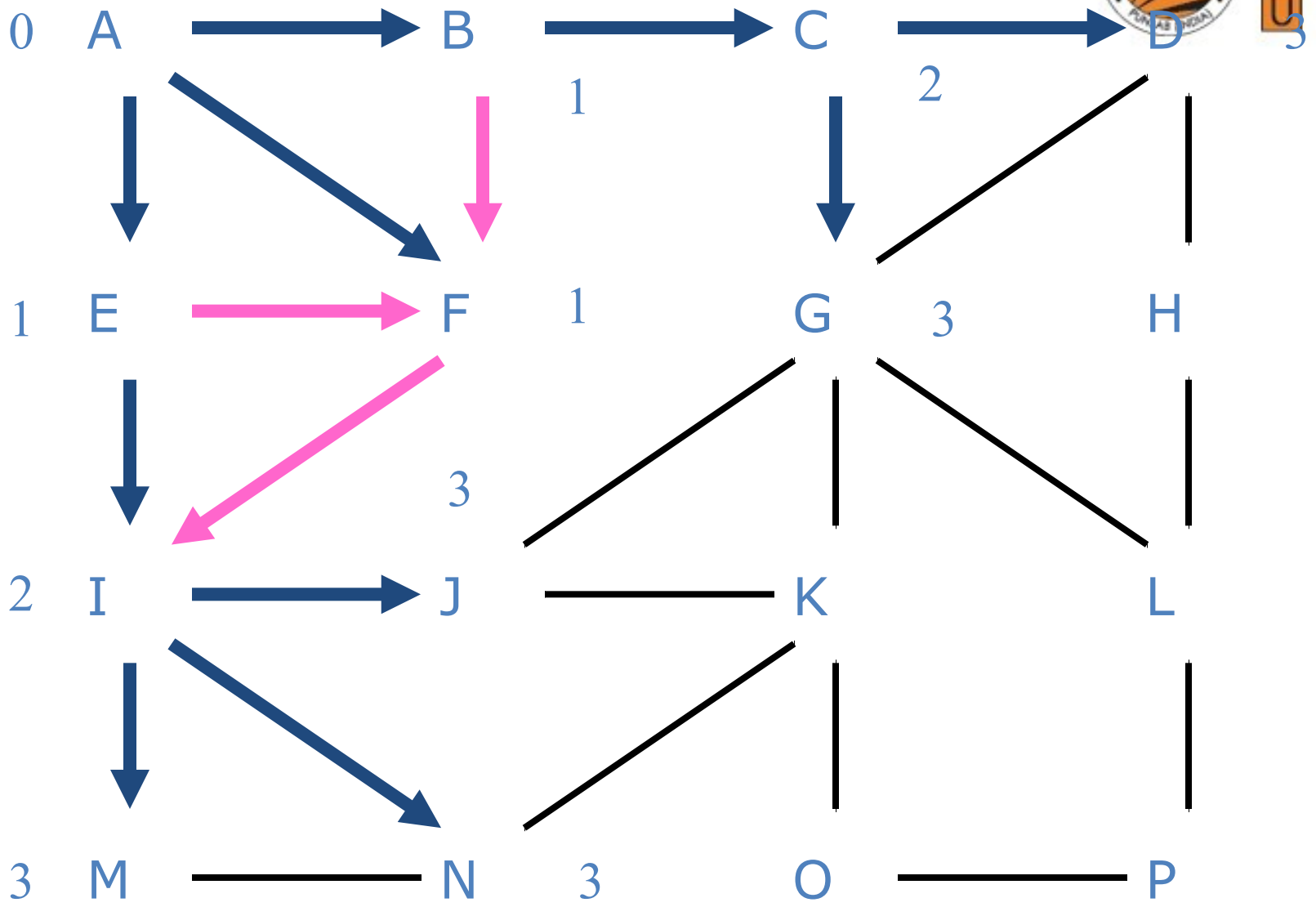
# An Example

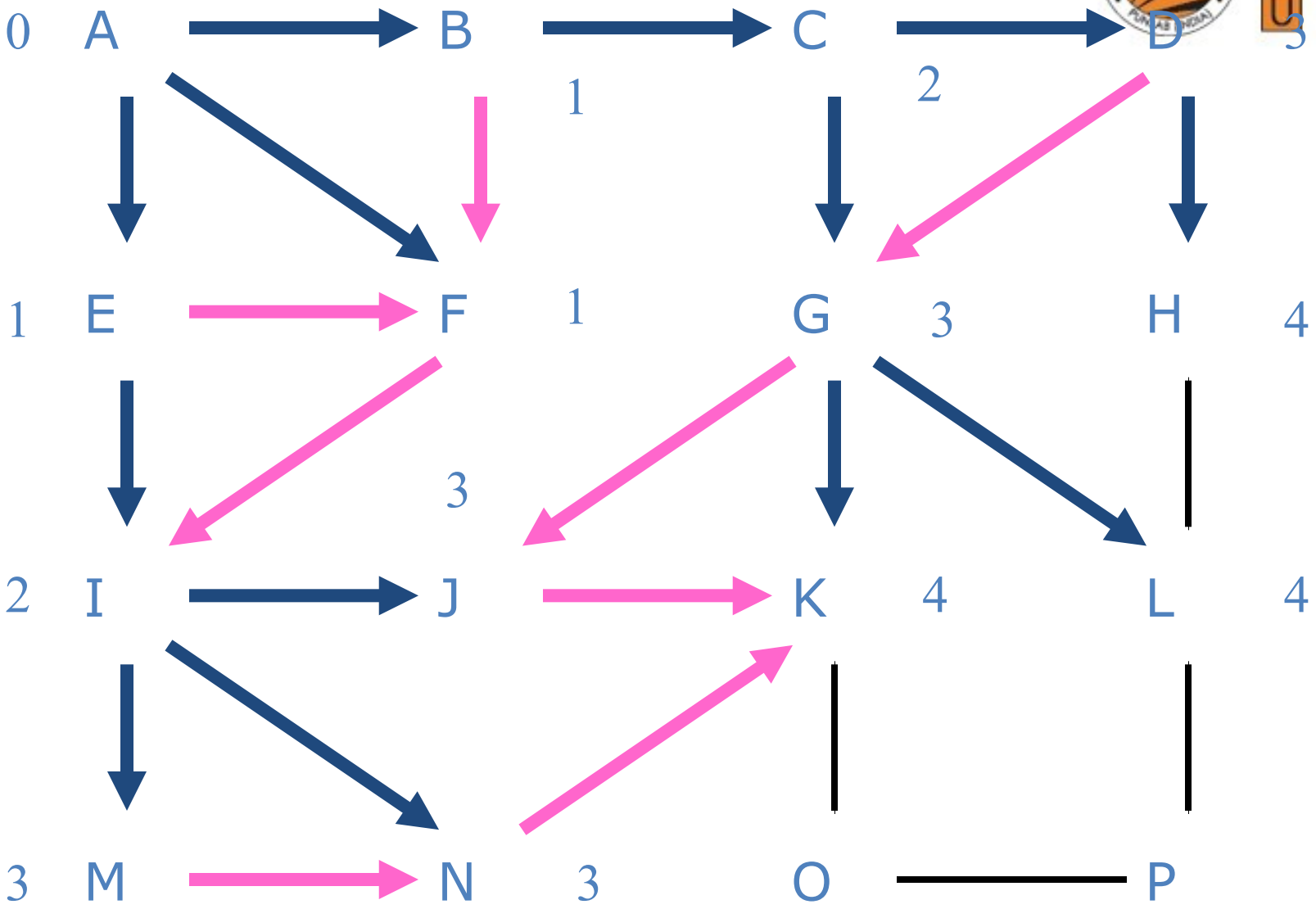


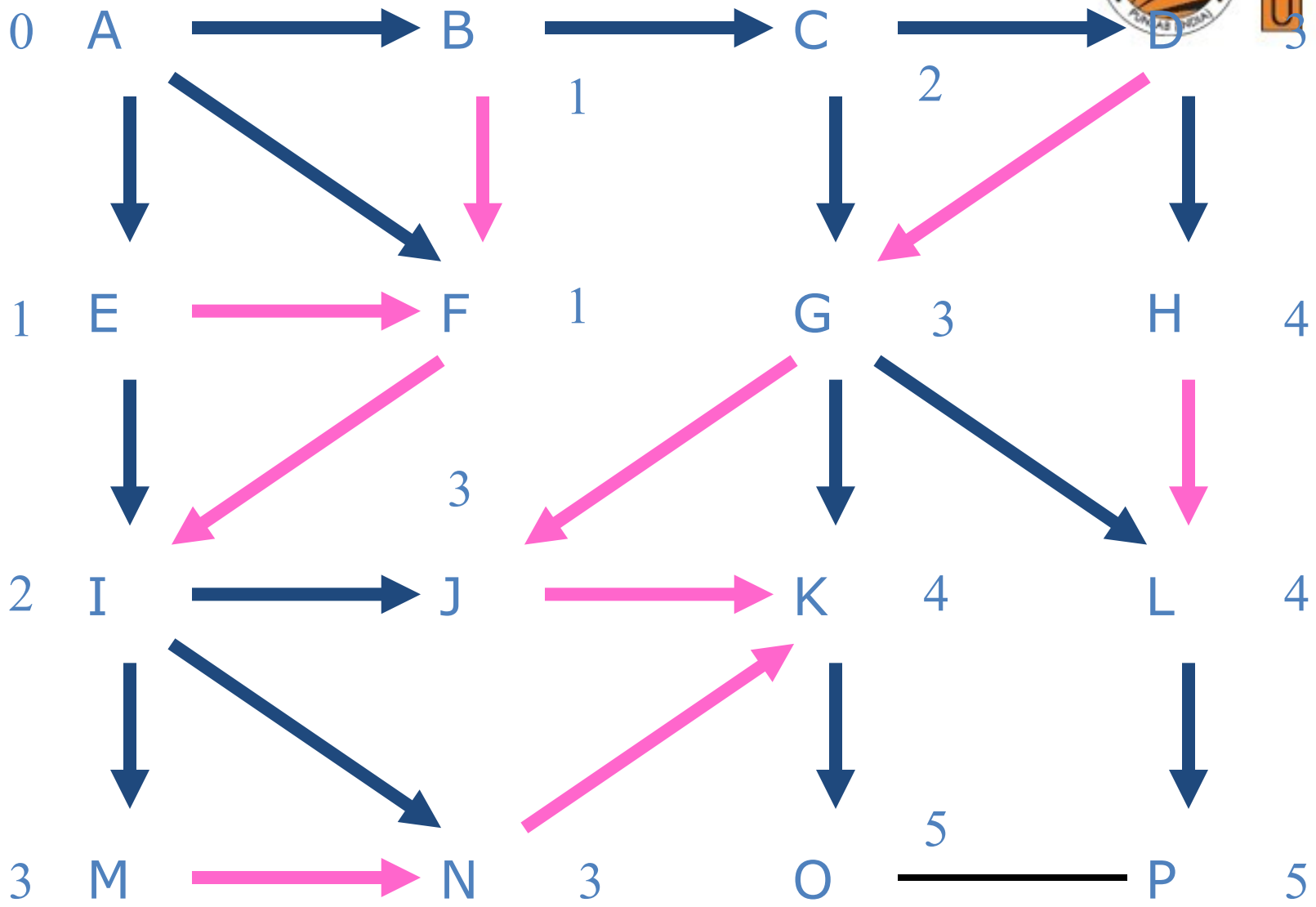

0
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P

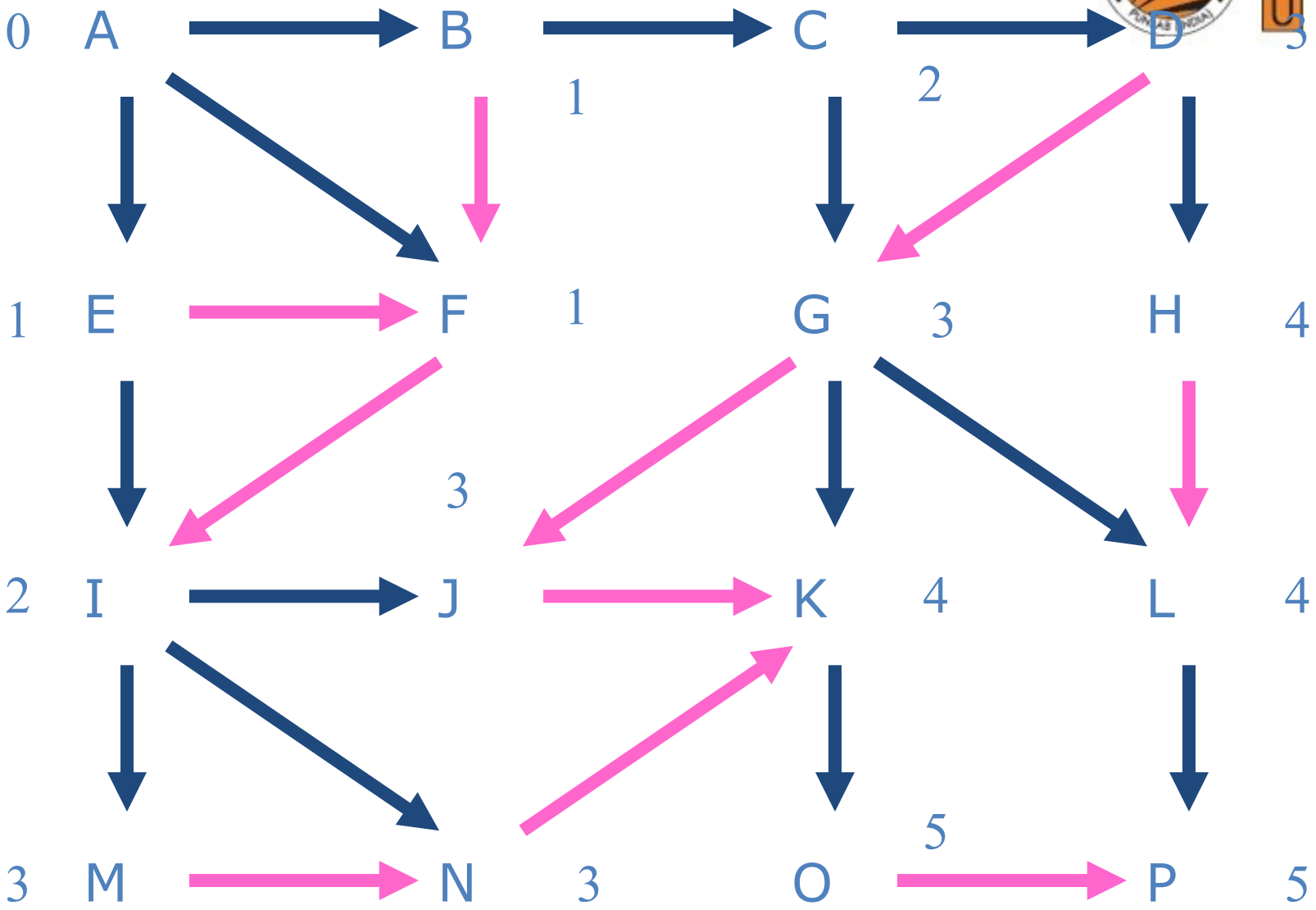

0
A
B
C
D
1
1
E
F
G
H
I
J
K
L
M
N
O
P

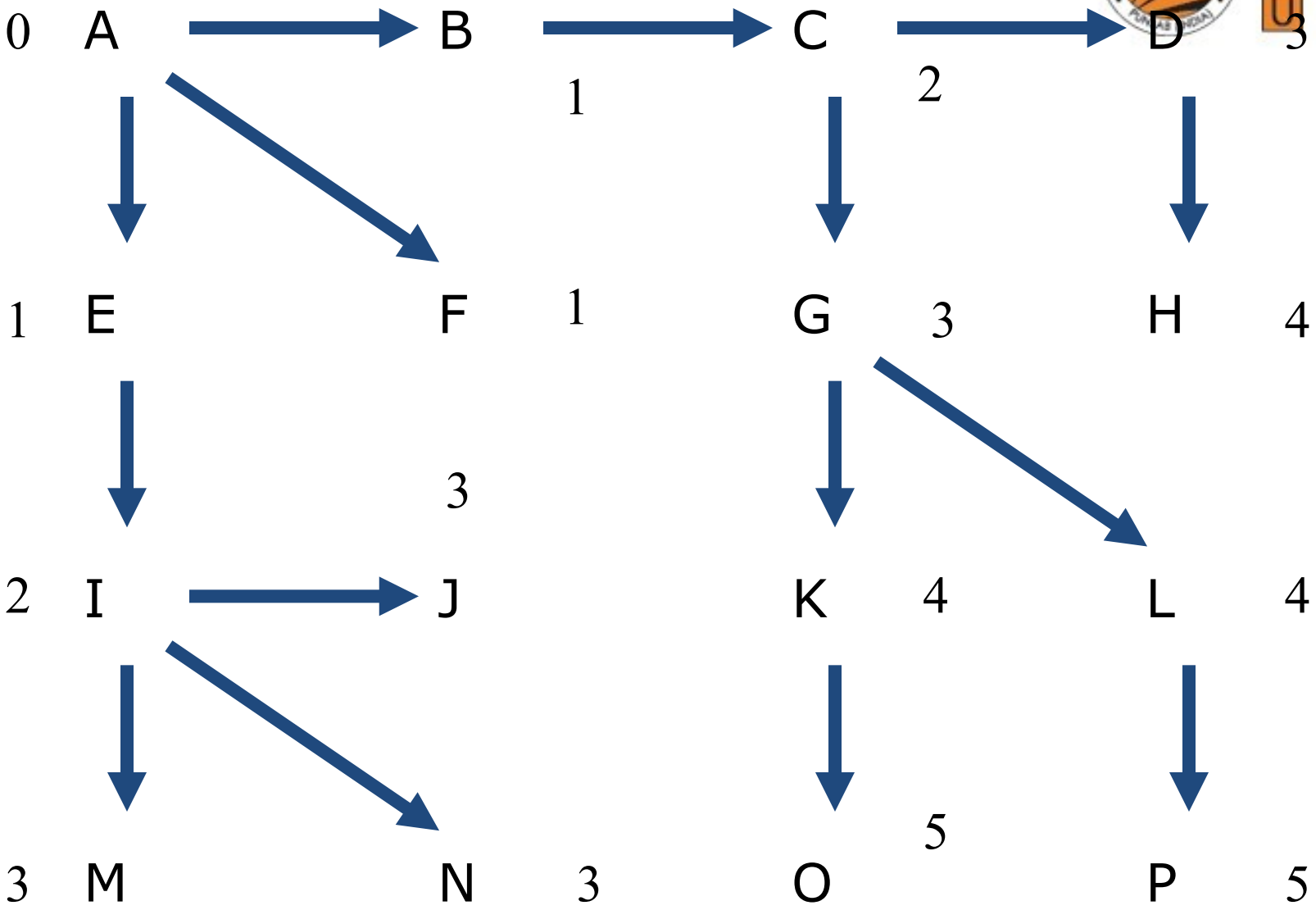

0
A
B
C
D
1
2
1
E
F
1
G
H
2
I
J
K
L
M
N
O
P


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


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


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

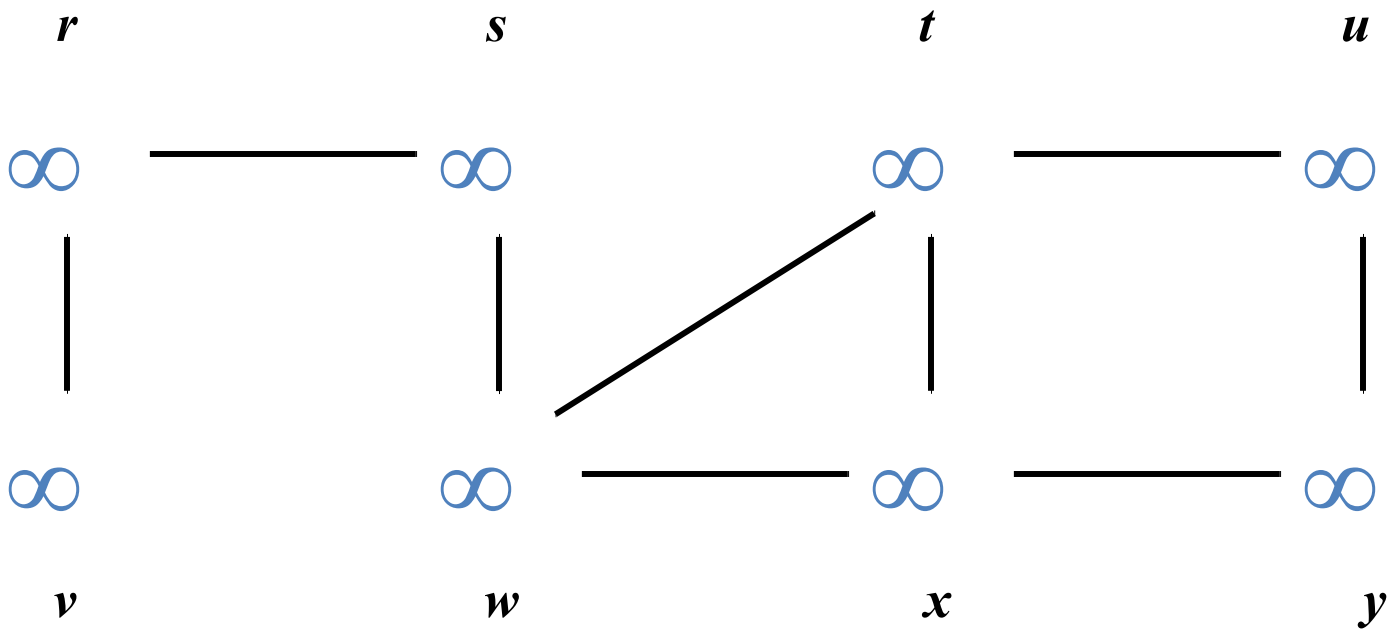

0 A
B
1
C
2
D
3
1 E
F
1
G
3
H
4
3
2 I
J
K
4
L
4
5
3 M
N
3
O
P
5

# BFS: Algorithm

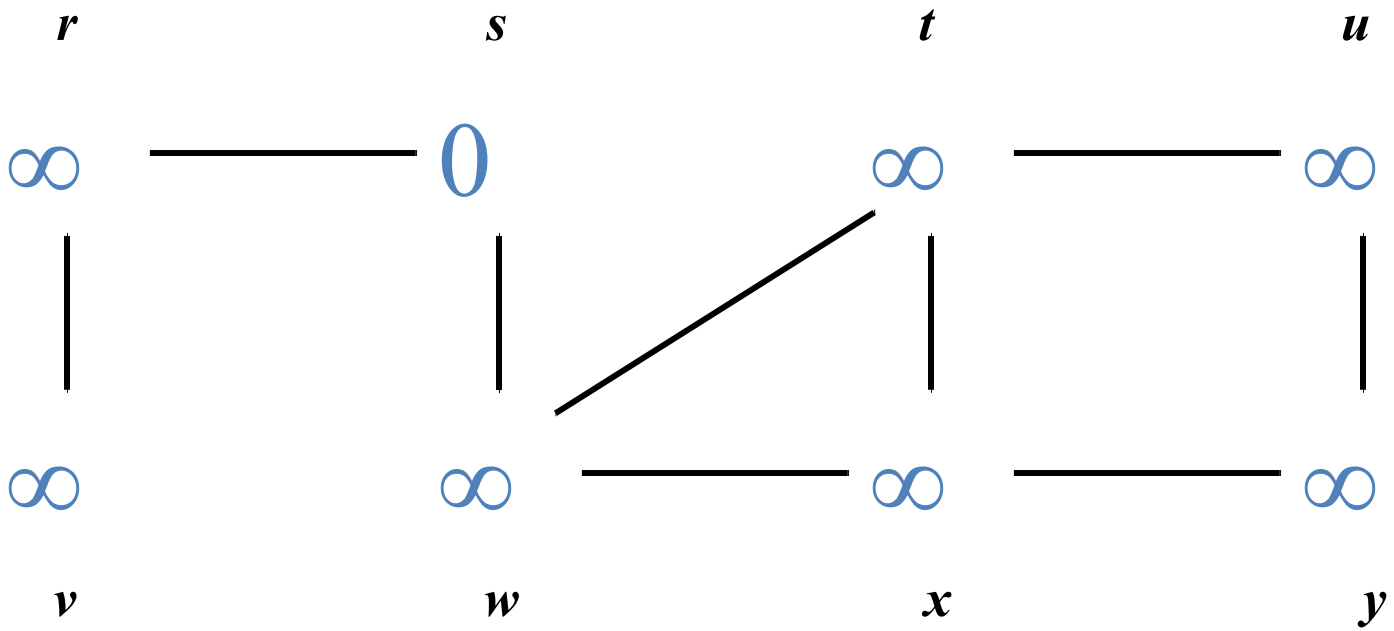
BFS(G, s)

- For each vertex u in V – {s},
  - color[u] = white;
  - d[u] = infinity;
  - $\pi$[u] = NIL
- color[s] = GRAY;  d[s] = 0;  $\pi$[s] = NIL;  Q = empty queue
- ENQUEUE(Q,s)
- while (Q not empty)
- u = DEQUEUE(Q)
- for each v $\in$ Adj[u]
- if color[v] = WHITE
- then color[v] = GREY
- d[v] = d[u] + 1; $\pi$[v] = u
- ENQUEUE(Q, v);
- color[u] = BLACK;
-

# Example

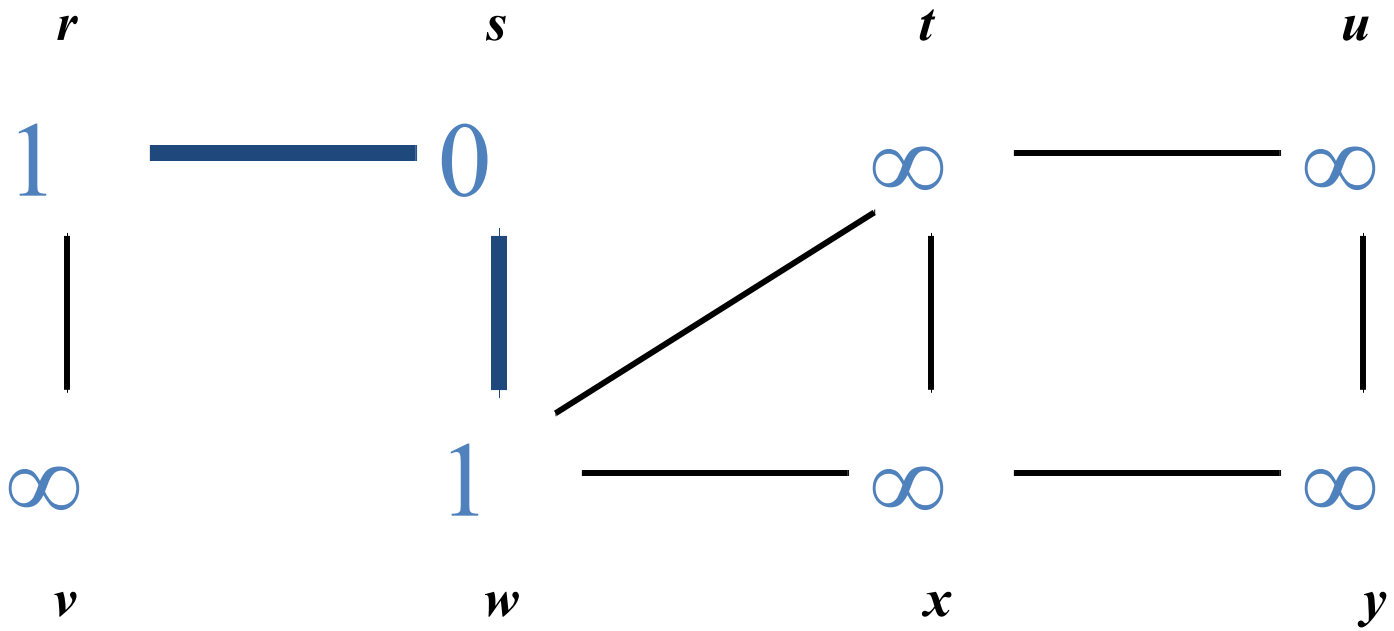


# Example



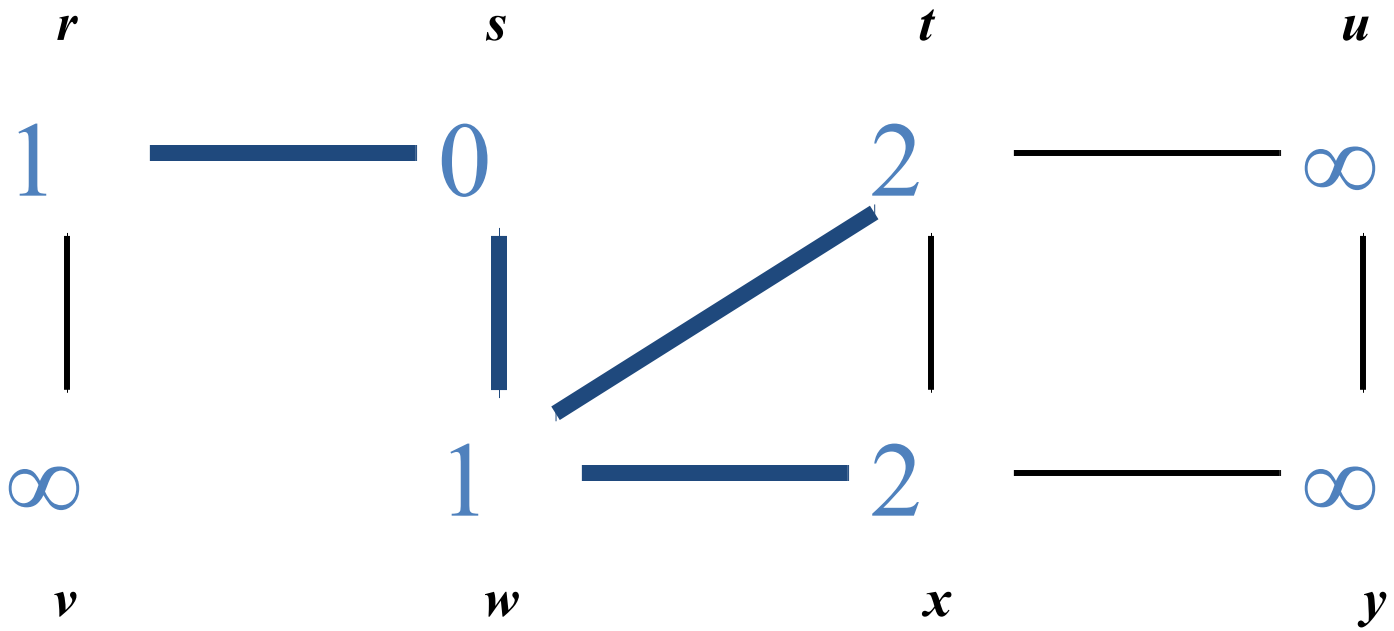

***Q:*** *s*

# Example



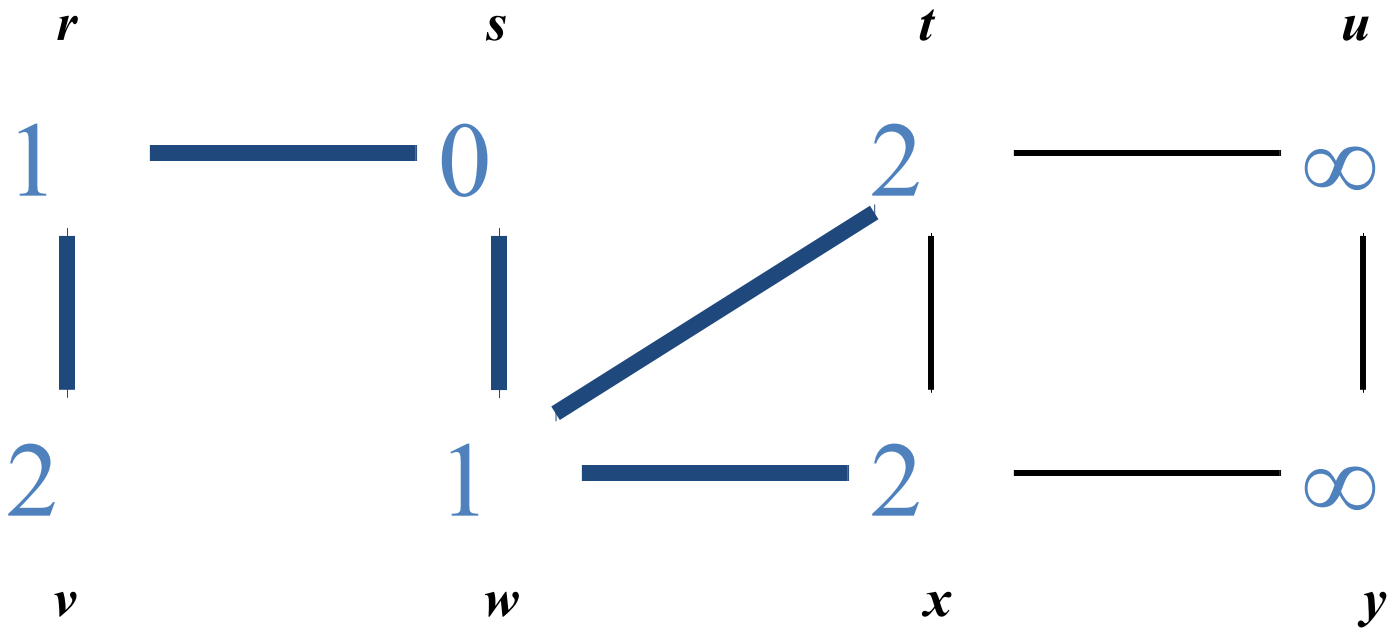

***Q:*** ***w*** ***r***

# Example



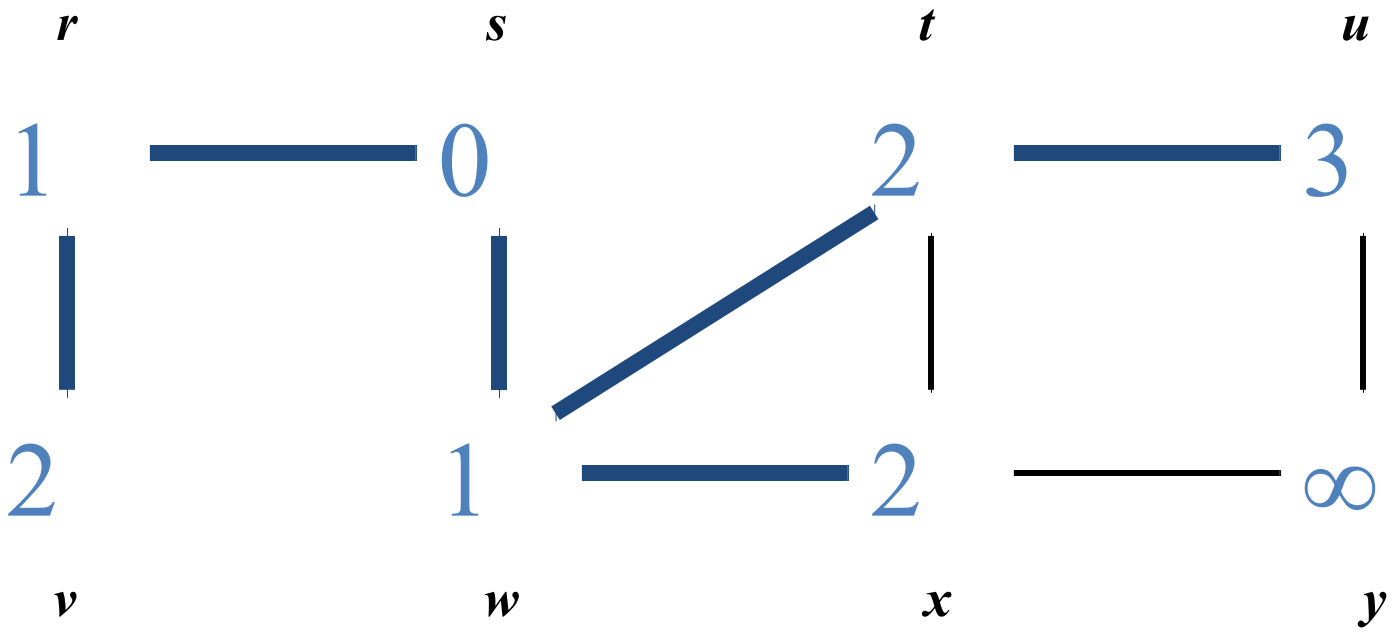

***Q:*** ***r*** ***t*** ***x***

# Example



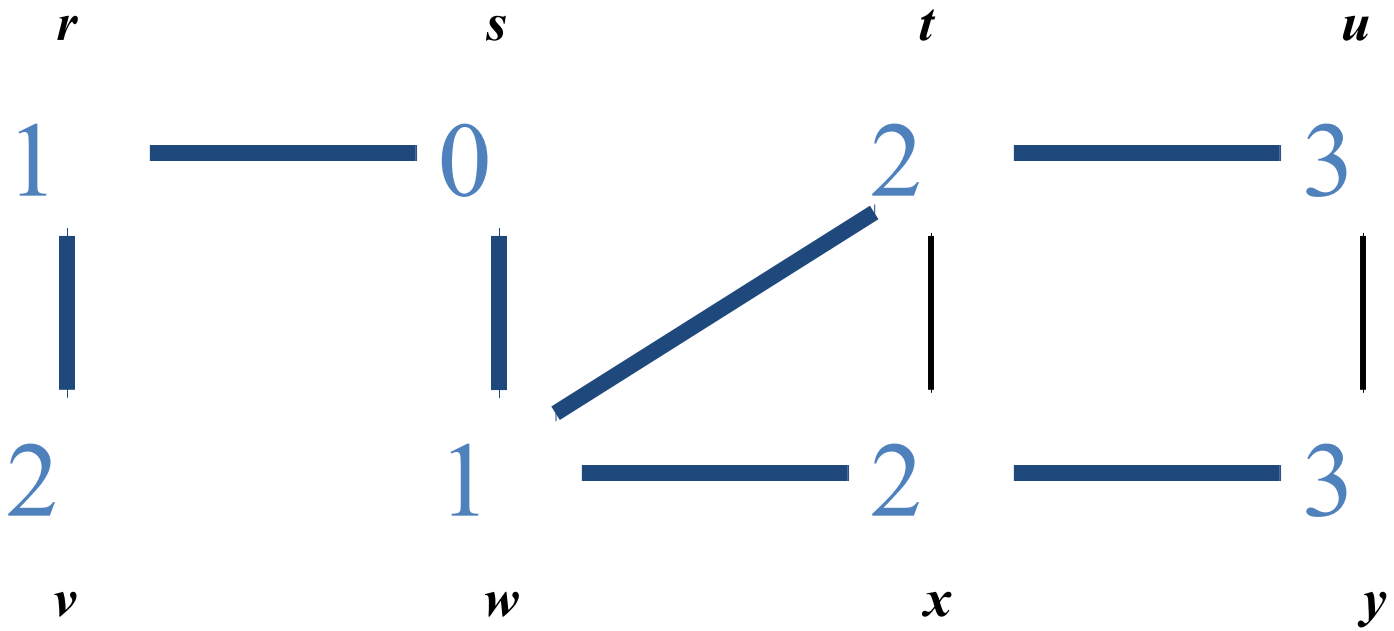

***Q:*** ***t*** ***x*** ***v***

# Example



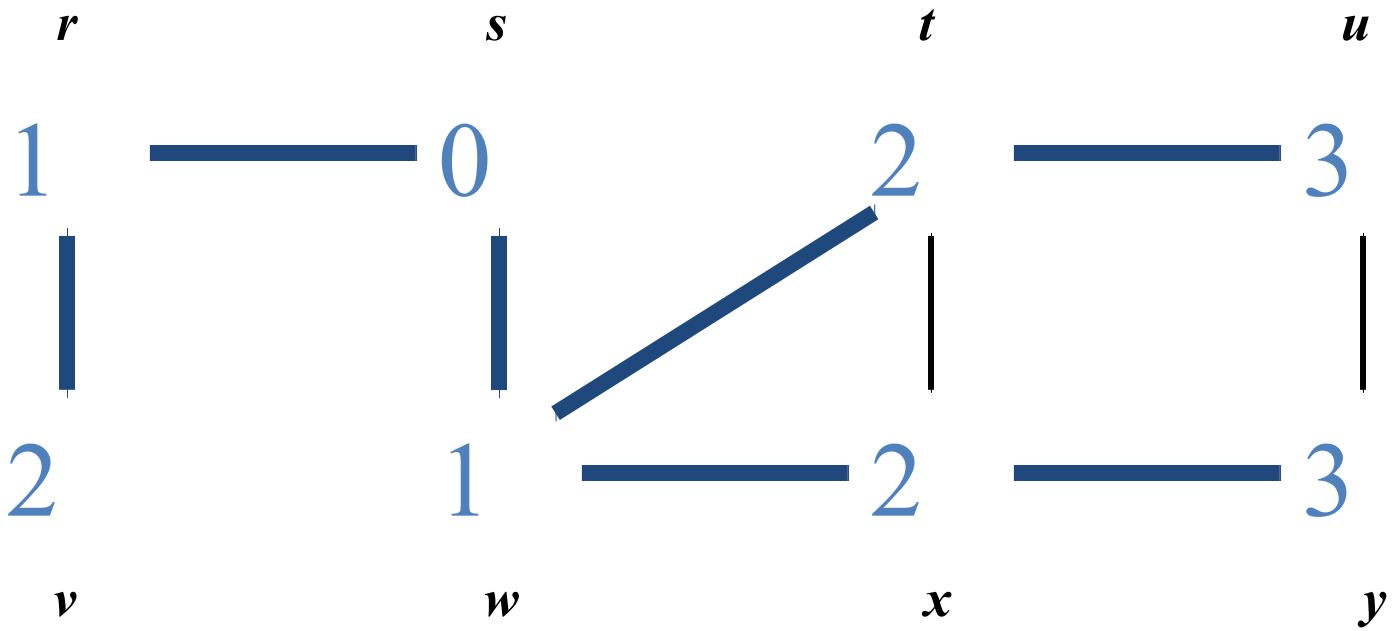

***Q:*** ***x*** ***v*** ***u***

# Example



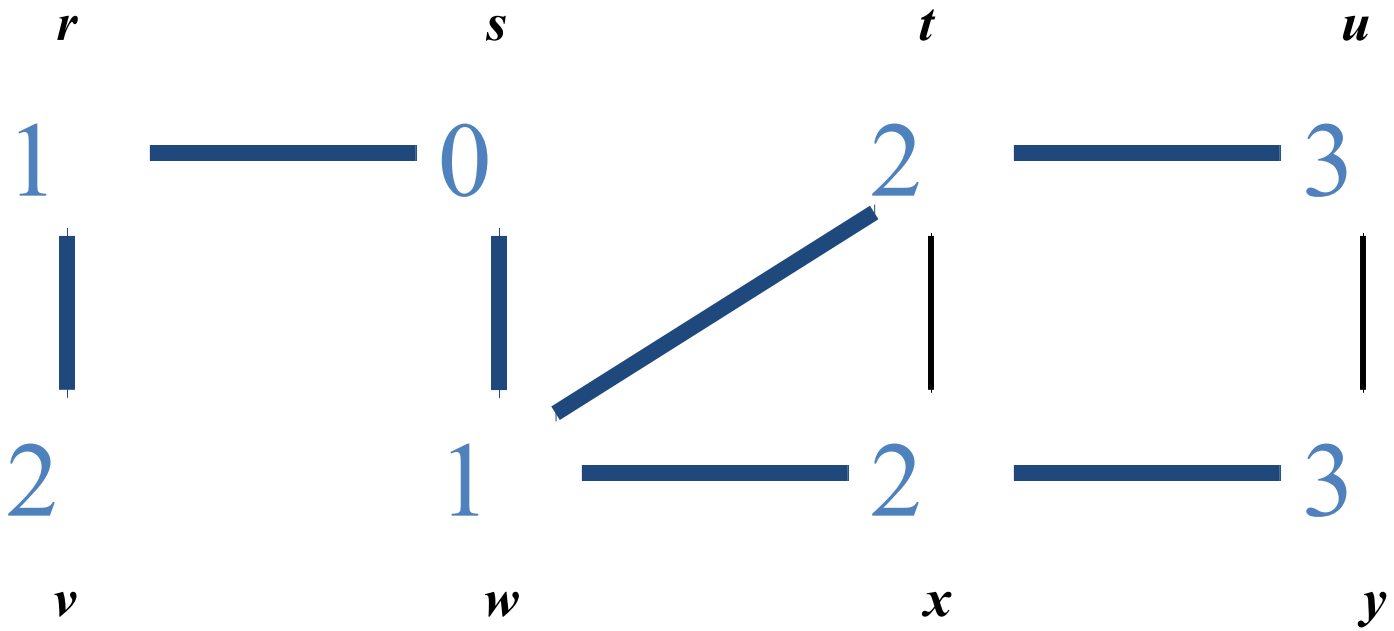

***Q:*** ***v*** ***u*** ***y***

# Example



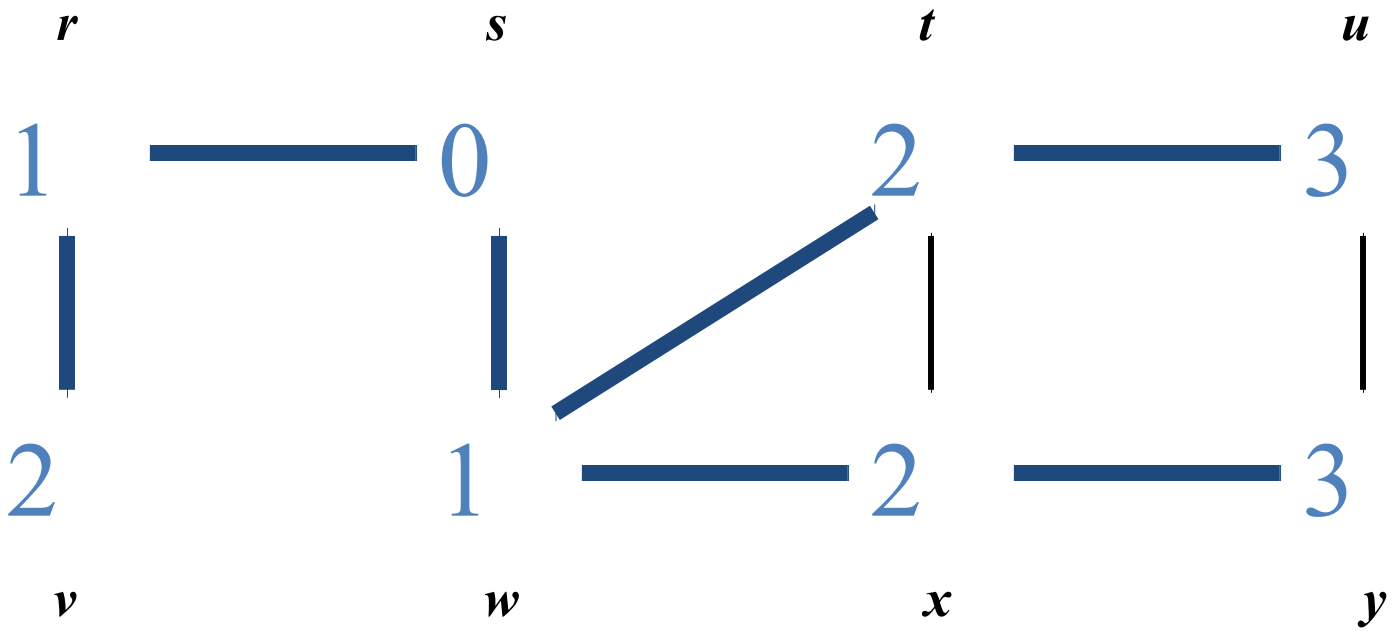

***Q:*** ***u*** ***y***

# Example




***Q:*** ***y***

# Example




***Q:*** ***Ø***

# BFS: Complexity Analysis


- Queuing time is O(V) and scanning all edges requires O(E)
- Overhead for initialization is O (V)
- So, total running time is O(V+E)

# BFS: Application


- Shortest path problem


L
P
U

Thank You !!!