CH08-320201 Algorithms and Data Structures

Lecture 20 — 24 Apr 2018

Prof. Dr. Michael Sedlmair

Jacobs University
Spring 2018

5. Graph Algorithms

5.1 Graph Representations

Directed and Undirected Graphs

Definition:

- A directed graph (digraph) $G=(V,E)$ is an ordered pair
- consisting of
  - a set V of vertices and
  - a set $E \subset V \times V$ of edges.
- In an undirected graph $G=(V,E)$, the edge set E consists of unordered pairs of vertices.

Number of edges and vertices

- In a graph, the number of edges is bound by $|E| = O(|V|^2)$.
- If G is connected, then $|E| \geq |V| - 1$.
- Hence, for a connected graph we get $\lg |E| = \Theta (\lg |V|)$.

Adjacency Matrices

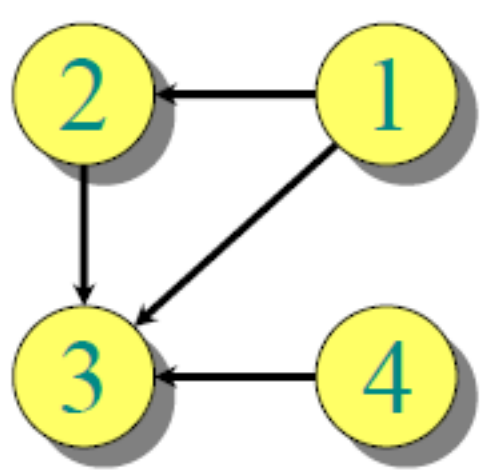
Definition:

- The adjacency matrix of a graph G=(V,E) with V={1,…,n} is the n×n matrix A given by

$$A[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E, \\ 0 & \text{if } (i,j) \notin E. \end{cases}$$

Dense representation: Storage requirements are $\Theta(|V|^2)$.

Example:



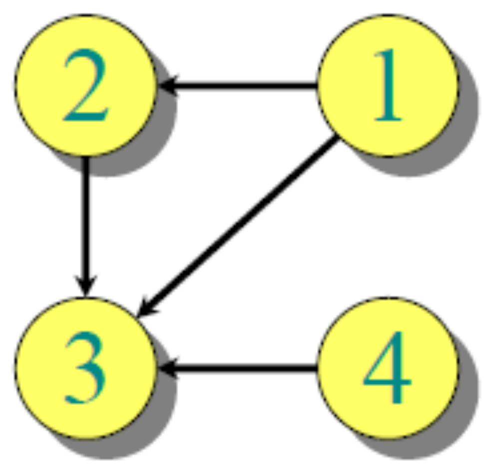
A	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0

Adjacency list

Definition:

- An adjacency list of a vertex v ϵ V is the list Adj[v] of vertices adjacent to v.

Example:

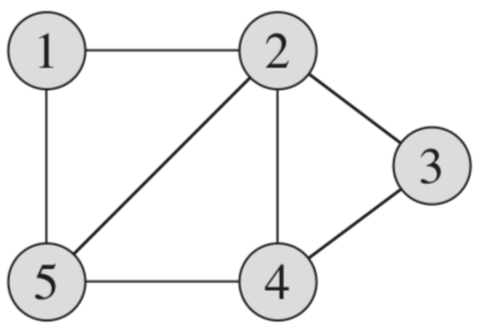


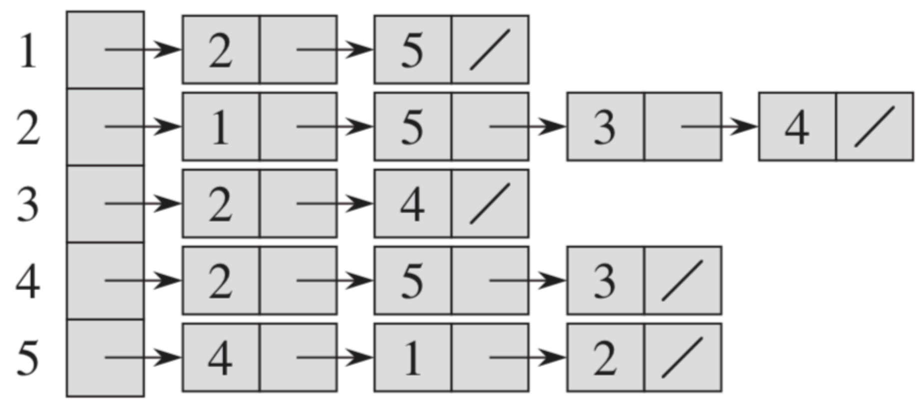
$$Adj[1] = \{2, 3\}$$
$$Adj[2] = \{3\}$$
$$Adj[3] = \{\}$$
$$Adj[4] = \{3\}$$

Sparse representation:

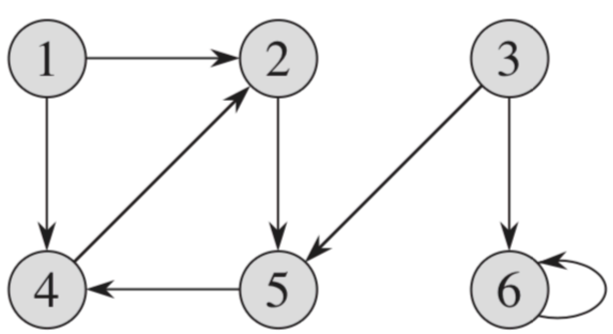
- Storage requirements for Adj[v] is Θ(|outgoing edges from v|).
- Storage requirement for Adj[v] for all v ϵ V is Θ(|E|).
- Overall storage requirement is Θ(|V|+|E|).

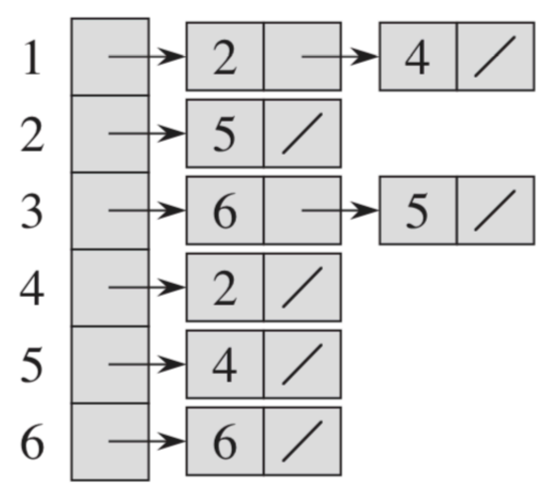
Examples for undirected & directed graphs





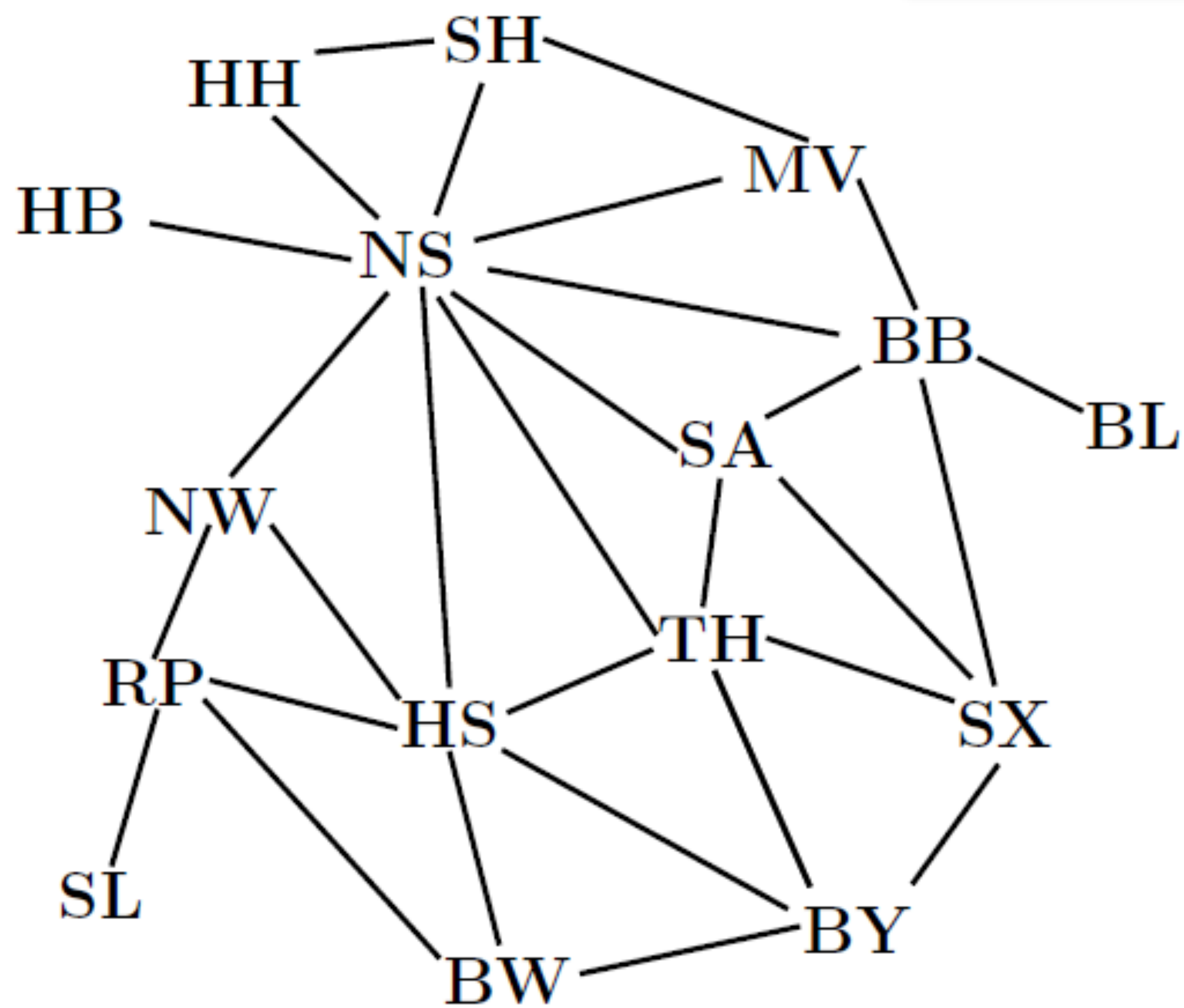
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

Application example: Neighboring states



5.2 Graph Searches

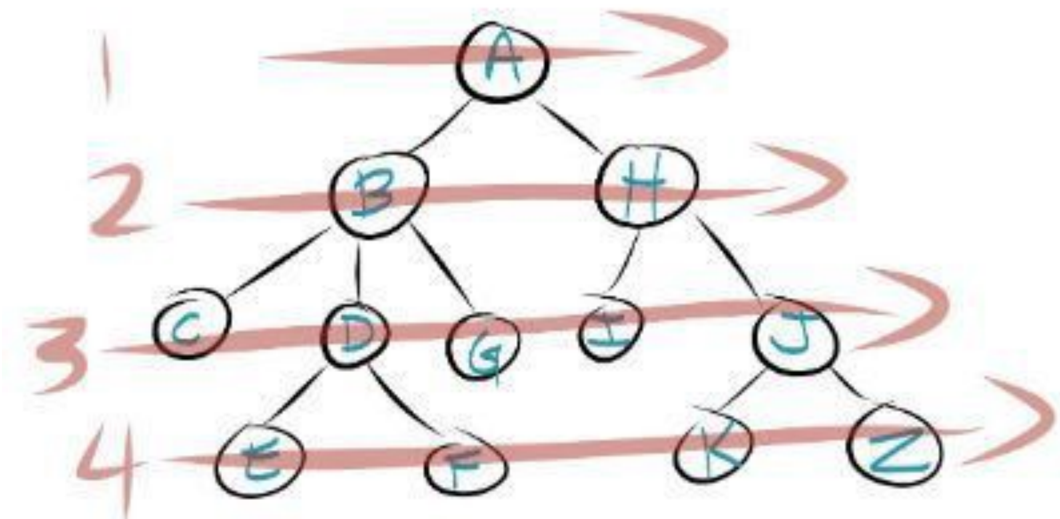
Breadth-first Search (BFS)

Problem:

- Given (directed or undirected) graph G = (V,E) and a starting vertex s ϵ V.
- Systematically explore all vertices reachable from s.

BFS strategy:

- First find all vertices of distance 1 from s, then of distance 2, then of distance 3, etc.

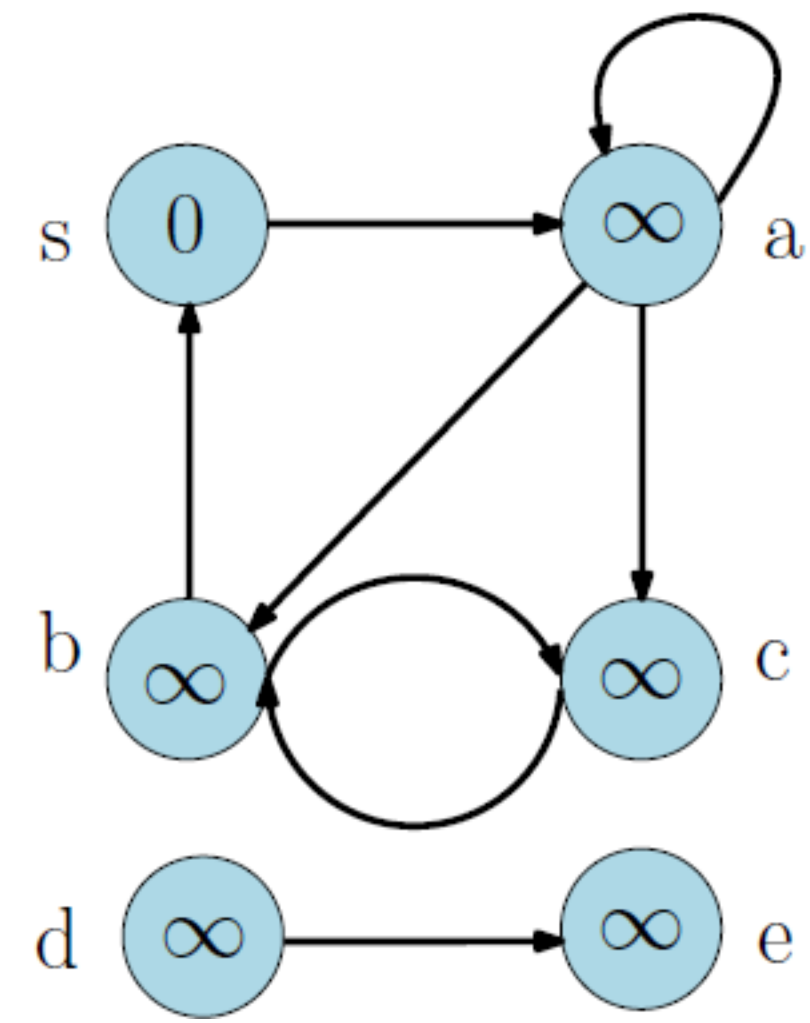


BFS approach

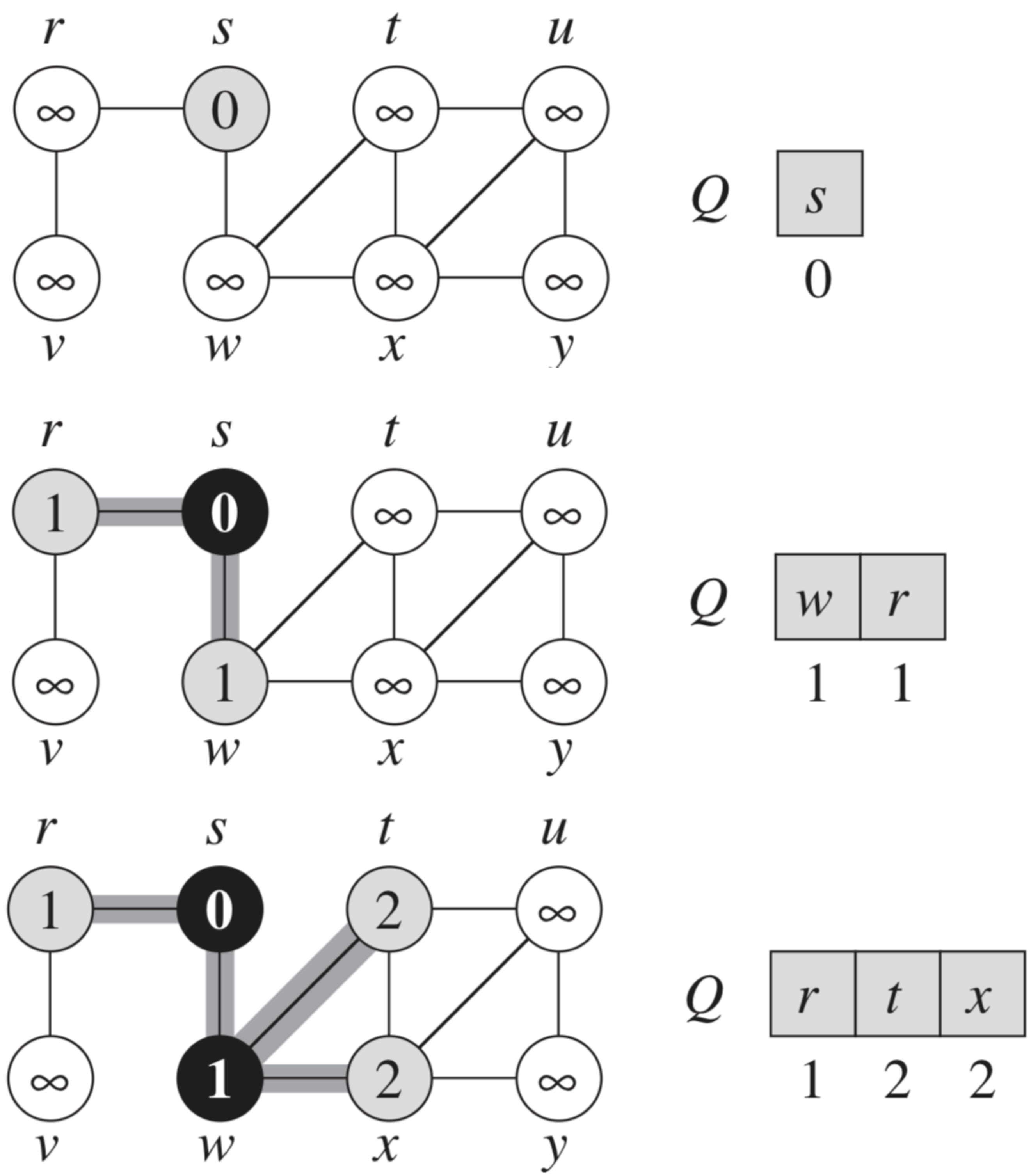
- Use adjacency-list representation.
- Use a color attribute for each vertex ∈ {white, gray, black}.
    - white: not detected yet
    - gray: just detected, waiting for us to explore their adjacency lists
    - black: done, all neighbors have been visited
- Store all gray vertices in a queue (FIFO principle).
- In addition, store for each vertex an attribute with the (topological) distance to starting vertex s.
- Finally, also store a pointer to the predecessor.

BFS algorithm

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

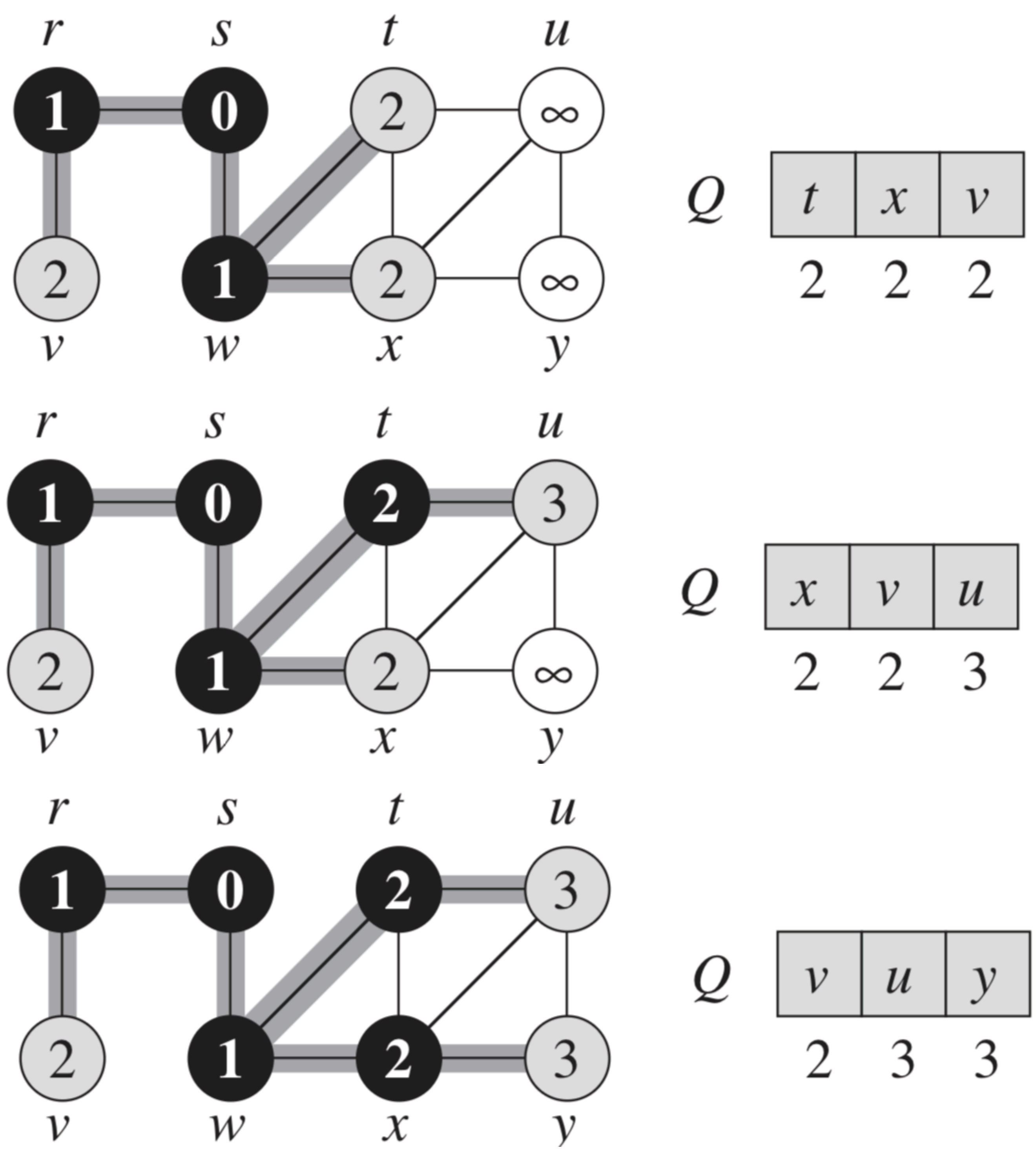


BFS example



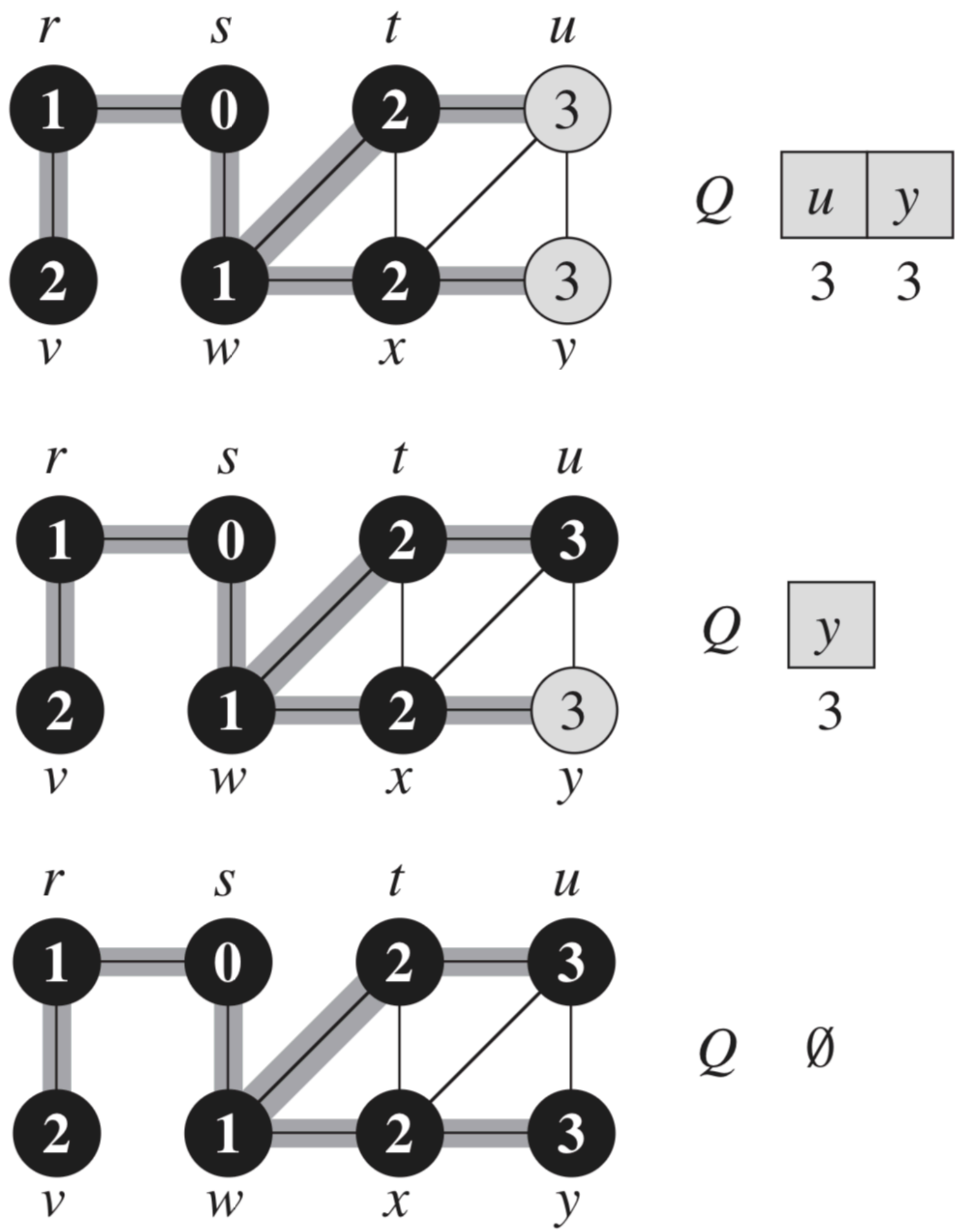

```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

BFS example




```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

BFS example




```
BFS(G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE(Q, s)
while Q ≠ ∅
    u = DEQUEUE(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

BFS analysis

- Each vertex is enqueued and dequeued once.
- Each queue operation is O(1).
- Total time for queue operations is O(|V|).
- Loop over adjacency list of all vertices is in total Θ(|E|).
- Together, we get a time complexity of O(|V|+|E|).

Breadth-first tree

- When storing the predecessors, we can construct the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ of G with

$$V_\pi = \{v \in V \mid v.\pi \neq NIL\} \cup \{s\}$$
$$E_\pi = \{(v.\pi, v) \mid v \in V_\pi - \{s\}\}$$

- This subgraph represents a tree structure.
- It is called the breadth-first tree.
- It contains a unique path from s to every vertex in V_π.
- All these paths are shortest paths in G.