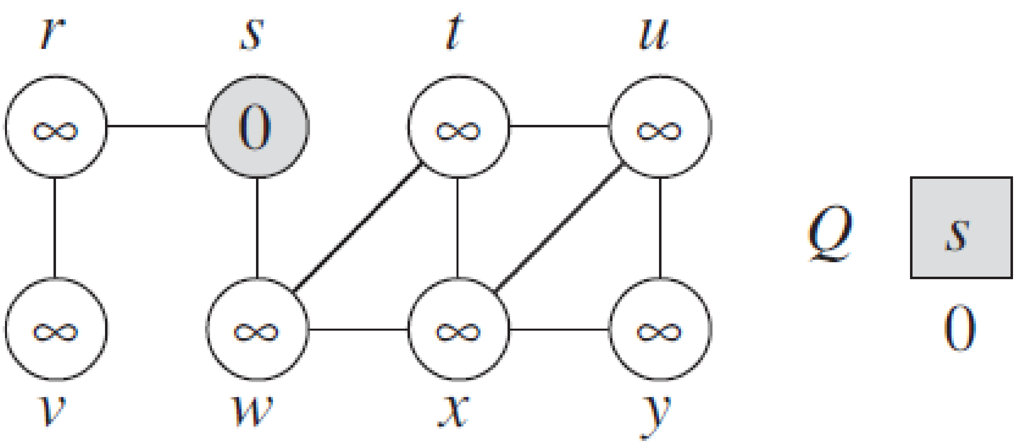
Breath First Search

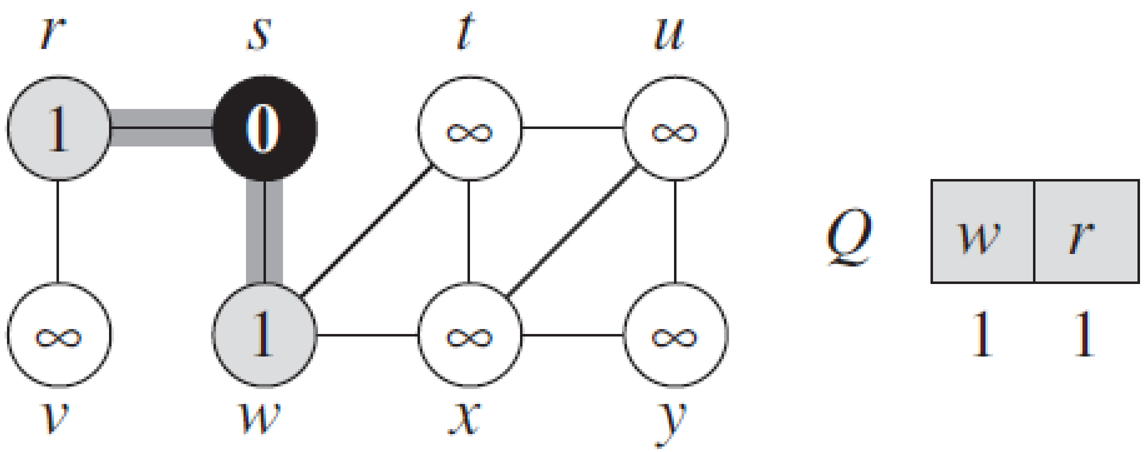
- Assume unweighted undirected graph

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 = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            Inject(Q,v)
    u.color = BLACK
```

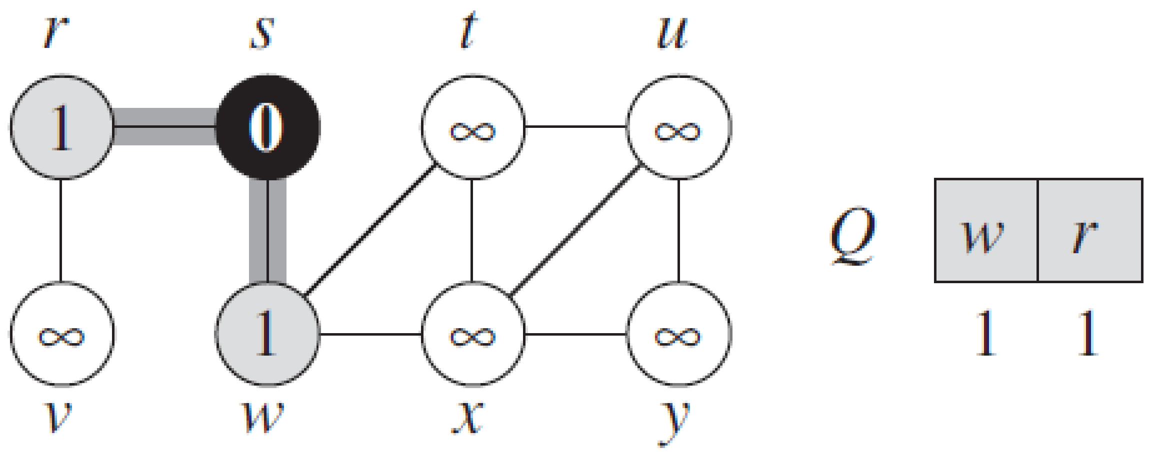
```
BFS(G,s)
For each vertex u ∈ G.V \ {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
     u = Eject(Q)
     for each v ∈ G.Adj[u]
          if v.color == WHITE
               v.color = GRAY
               v.d = u.d + 1
  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

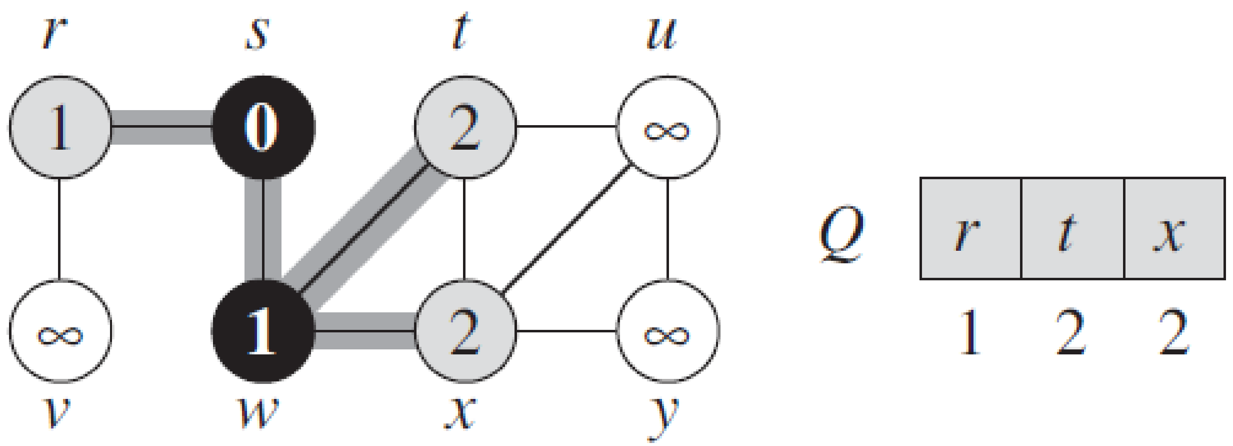




BFS(G,s)

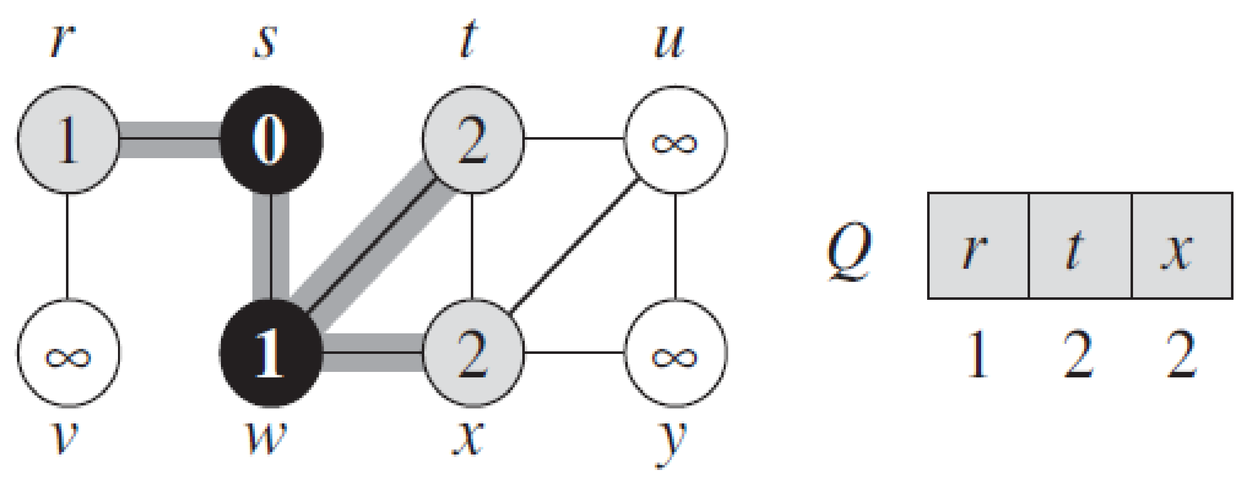
```
For each vertex u ∈ G.V \ {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
 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

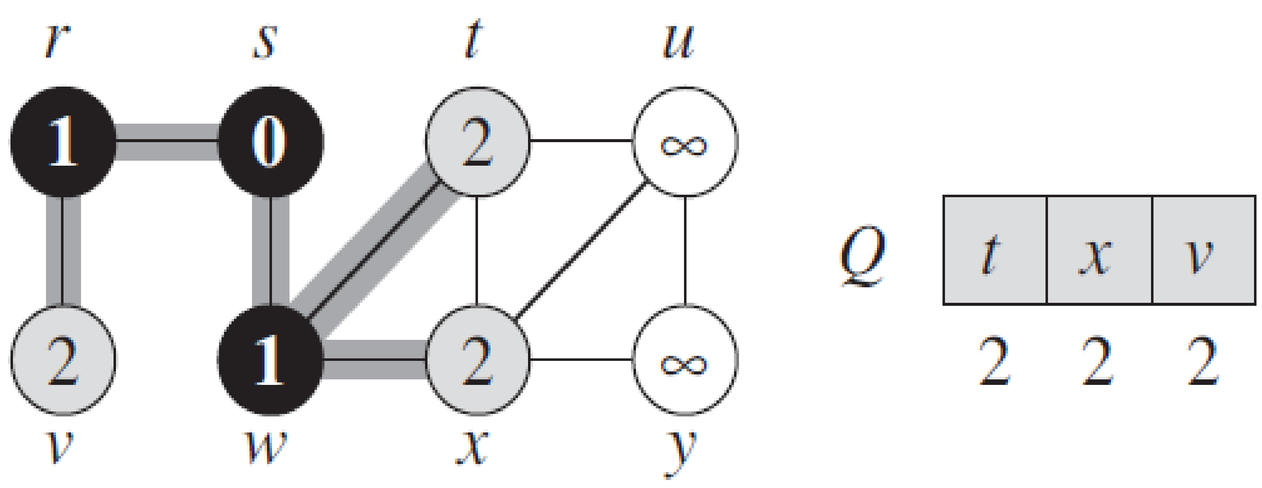




BFS(G,s)

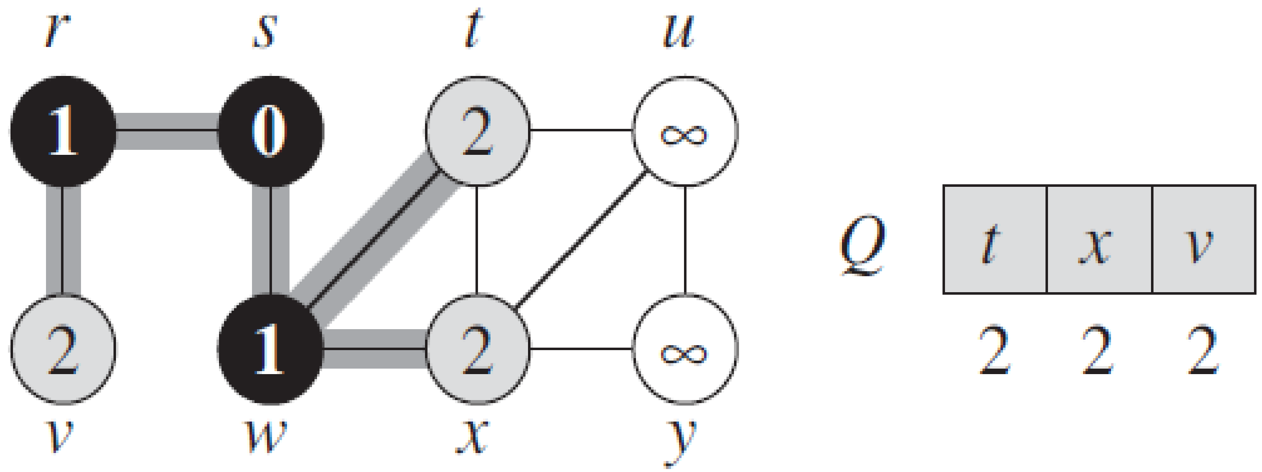
```
For each vertex u ∈ G.V \ {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            Inject(Q,v)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

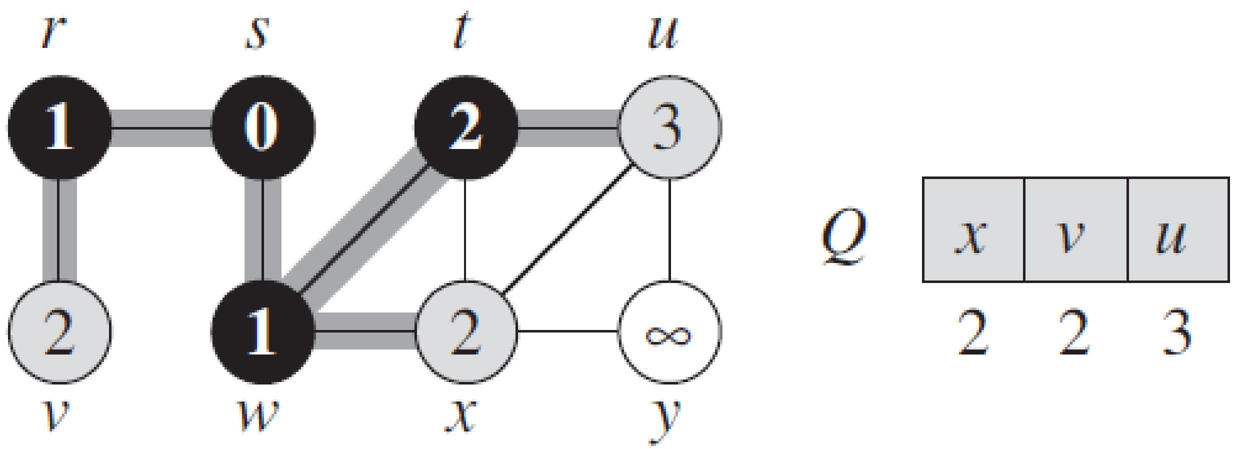




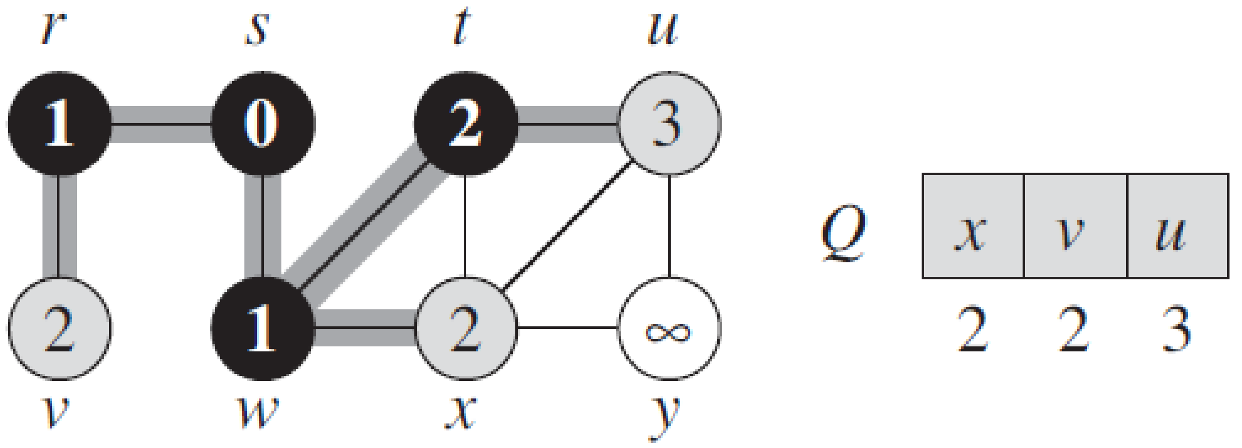
BFS(G,s)

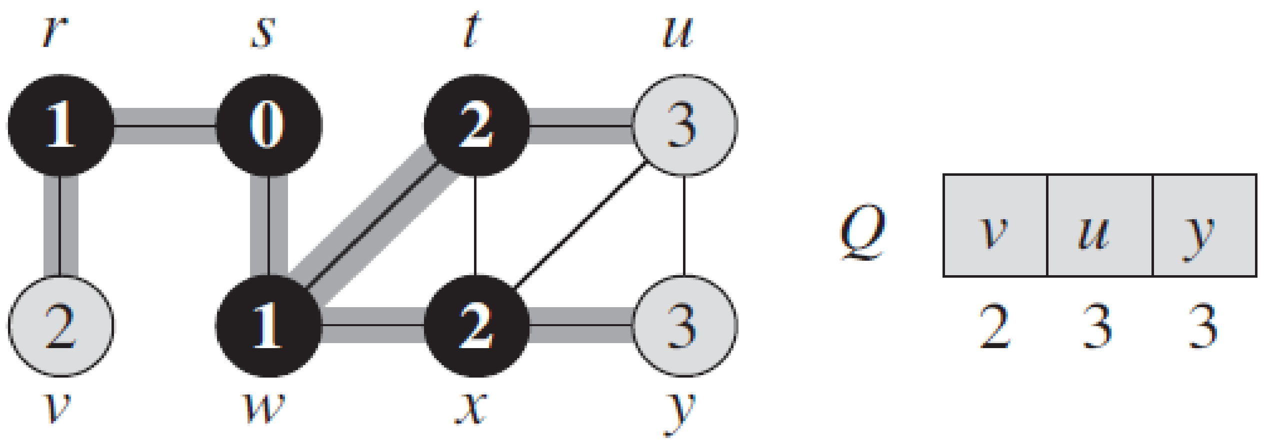
```
For each vertex u ∈ G.V \ {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            Inject(Q,v)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



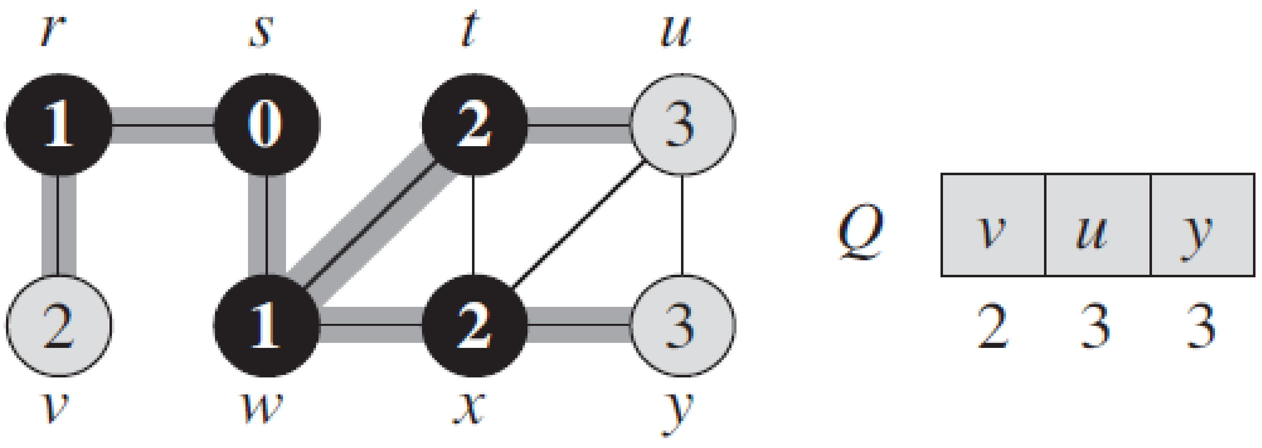


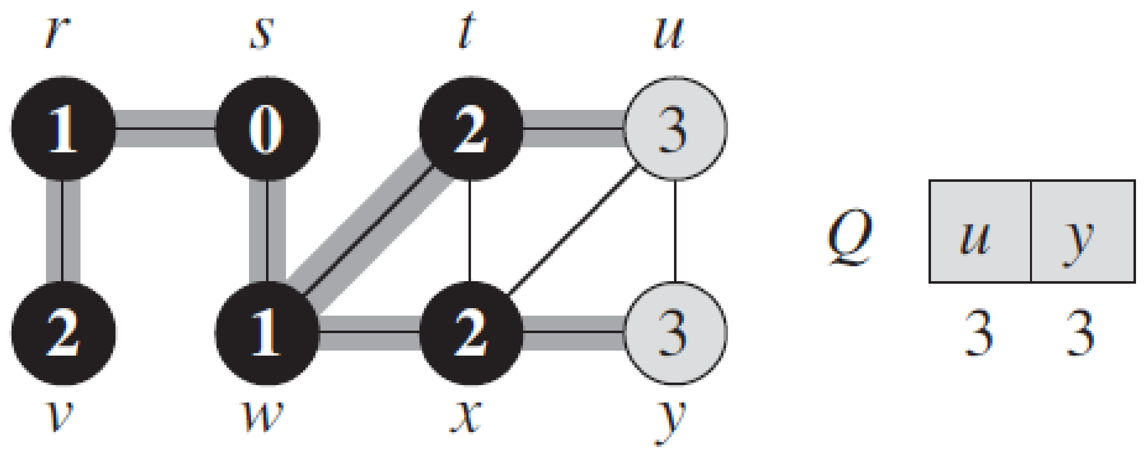
```
BFS(G,s)
For each vertex u ∈ G.V \ {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π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



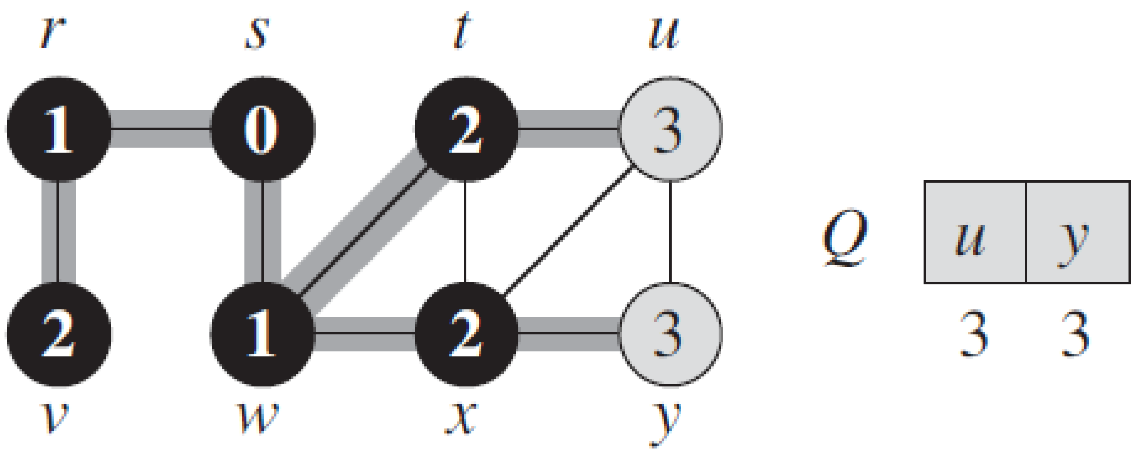


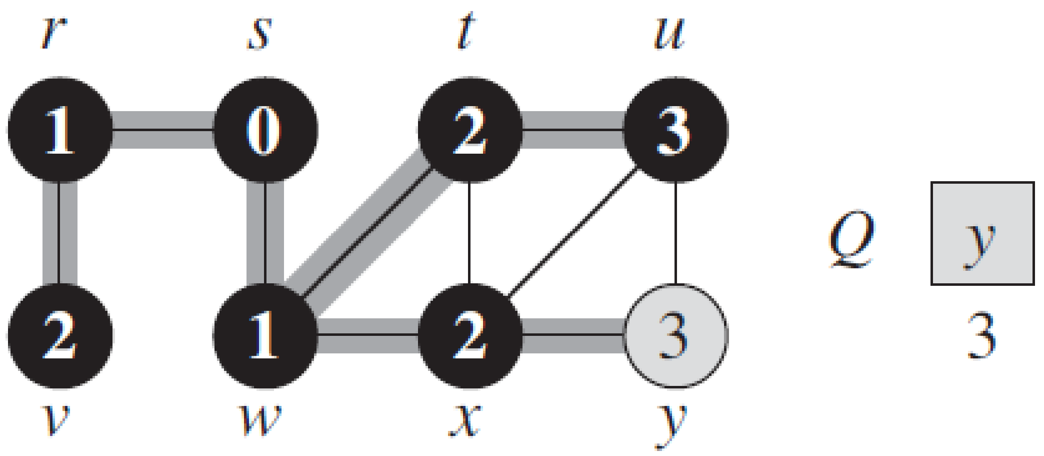
```
BFS(G,s)
For each vertex u ∈ G.V \ {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π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



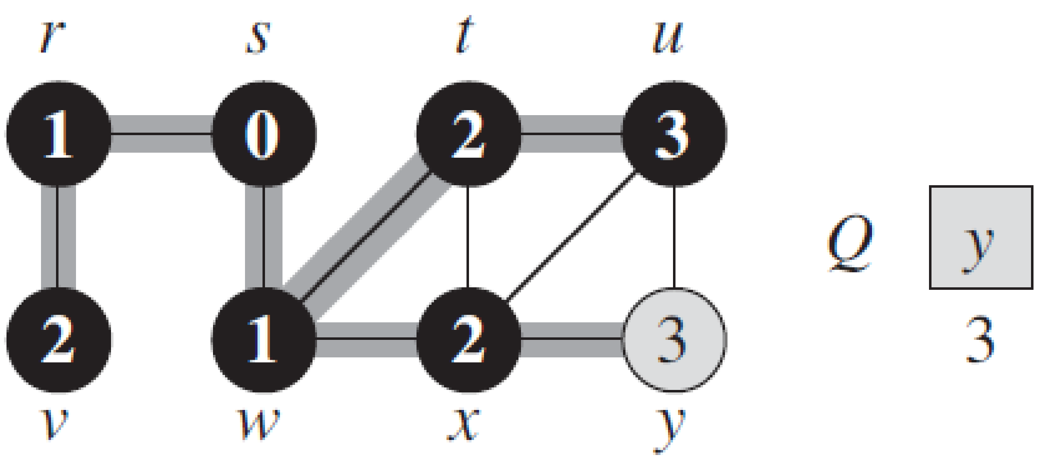


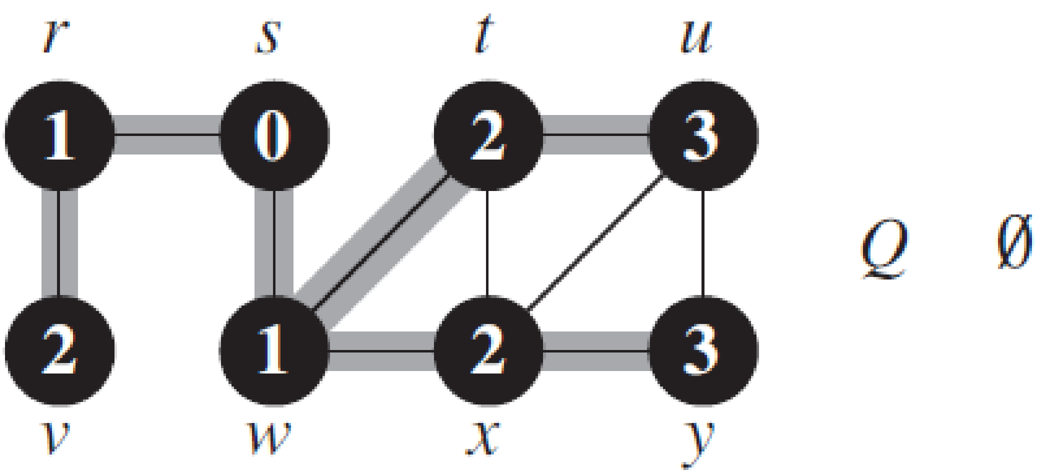
```
BFS(G,s)
For each vertex u ∈ G.V \ {s}
      u.color = WHITE
      u.d = ∞
      u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u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





```
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 = ∅
Inject(Q,s)
while Q ≠ ∅
    u = Eject(Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            Inject(Q,v)
    u.color = BLACK
```





BFS(G,s)

For each vertex $u \in G.V \setminus \{s\}$
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
$Q = \emptyset$
Inject(Q,s)
while $Q \neq \emptyset$
    u = Eject(Q)
    for each $v \in G.Adj[u]$
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            Inject(Q,v)
    u.color = BLACK

Running time: $O(n + m)$

Inv1: A vertex v is GRAY iff it is in Q

Inv2: If $v_1, v_2, \ldots, v_r$ are in Q then

$$v_1.d \leq v_2.d \leq \cdots \leq v_r.d \leq v_1.d + 1$$

Cor2: If v_i gets into Q before v_j then

$$v_i.d \leq v_j.d$$

BFS computes shortest (in #edges) paths from s

- Let $\delta(s, v)$ be the length of the shortest path from s to v, $\delta(s, s) = 0$, $\delta(s, v) = \infty$ if v is not reachable from s.

- Thm (BFS):
  1. When BFS terminates $\forall v: v.d = \delta(s, v)$.
  2. If $\delta(s, v) < \infty$ then there is a shortest path from s to v that ends with $(v.\pi, v)$

BFS tree

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

- Then (by the BFS Thm) $G_\pi = (V_\pi, E_\pi)$ contains a shortest path from s to every $v \in V_\pi$.
- Since G_π is connected and $|E_\pi| = |V_\pi| - 1$ ➔ it is a tree.