DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

BREADTH-FIRST SEARCH

AGENDA

- Breadth-first search algorithm
  - Adjacency list
  - Queue
  - Running time

GRAPH SEARCHING ALGORITHMS

- The graph searching algorithms systematically follow the edges of the graph so as to visit the vertices of the graph.
- Also used to discover structural information about a given graph.

- **Two algorithms**
  - Breadth-first search (BFS)
  - Depth-first search (DFS)

BREADTH-FIRST SEARCH (BFS)

- Given a graph $G = (V, E)$ and a **distinguished source vertex s**, breadth-first search systematically **explores the edges of G** to "discover" every vertex that is reachable from s.

- Compute the distance from s to each reachable vertex.

- Produce a "breadth-first tree" with root s that contains all reachable vertices.
  - For any vertex v reachable from s, the simply path in the breadth-first tree from s to v corresponds to a "shortest path" from s to v in G.

THE BFS ALGORITHM

- Input
  - a graph $G = (V, E)$ that is represented BY **adjacency lists**
  - the source node s
- Data structure
  - A first-in, first-out queue Q
    - Operations
      - ENQUEUE
      - DEQUEUE

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$u.color =$ WHITE
$u.d = \infty$
$u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$u =$ DEQUEUE (Q)
for each $v \in G.Adj[u]$
if $v.color ==$ WHITE
$v.color =$ GRAY
$v.d = u.d + 1$
$v.\pi = u$
ENQUEUE (Q, v)
$u.color =$ BLACK

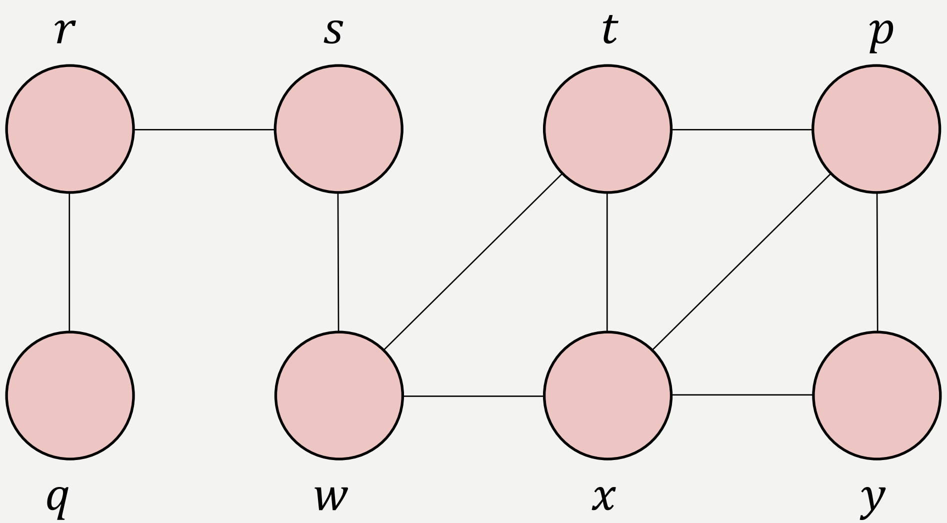
THE BFS ALGORITHM
THE VERTEX OBJECT

- The algorithm attaches several **additional attributes** to each vertex in the graph.
- For each vertex $u \in V$,
  - **$u.color$** – distinguish between *discovered* and *undiscovered* vertices.
    - The color could be, **BLACK**, **GRAY** or **WHITE**.
  - **$u.\pi$** – the ***predecessor*** of vertex u.
    - If u has no predecessor, then $u.\pi = \mathtt{NIL}$.
  - **$u.d$** – the ***distance*** from the source s to vertex u computed by the 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
```

THE BFS ALGORITHM
IN ACTION

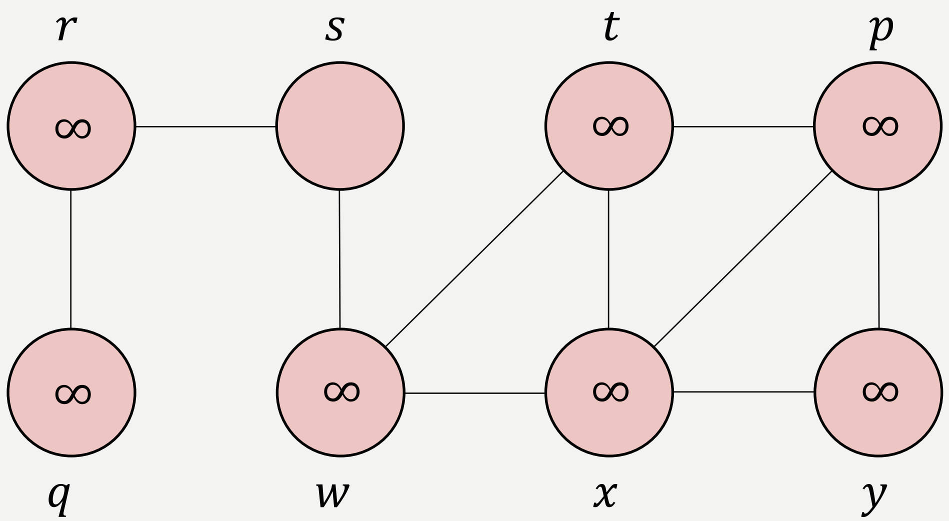
- Apply the algorithm on the graph below.




```
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
```

THE BFS ALGORITHM
IN ACTION

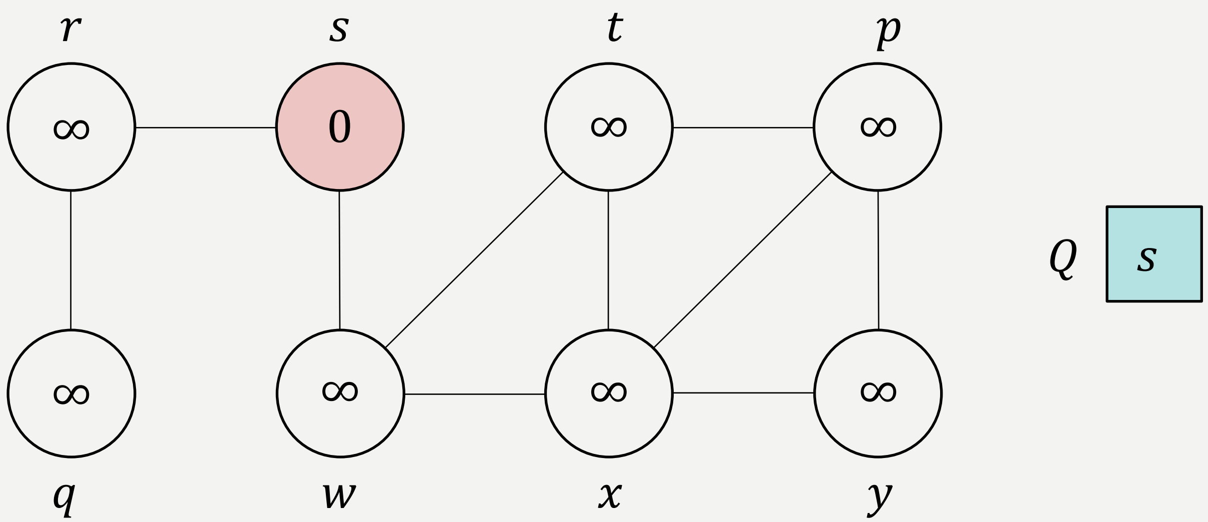
- Apply the algorithm on the graph below.




```
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
```

THE BFS ALGORITHM
IN ACTION

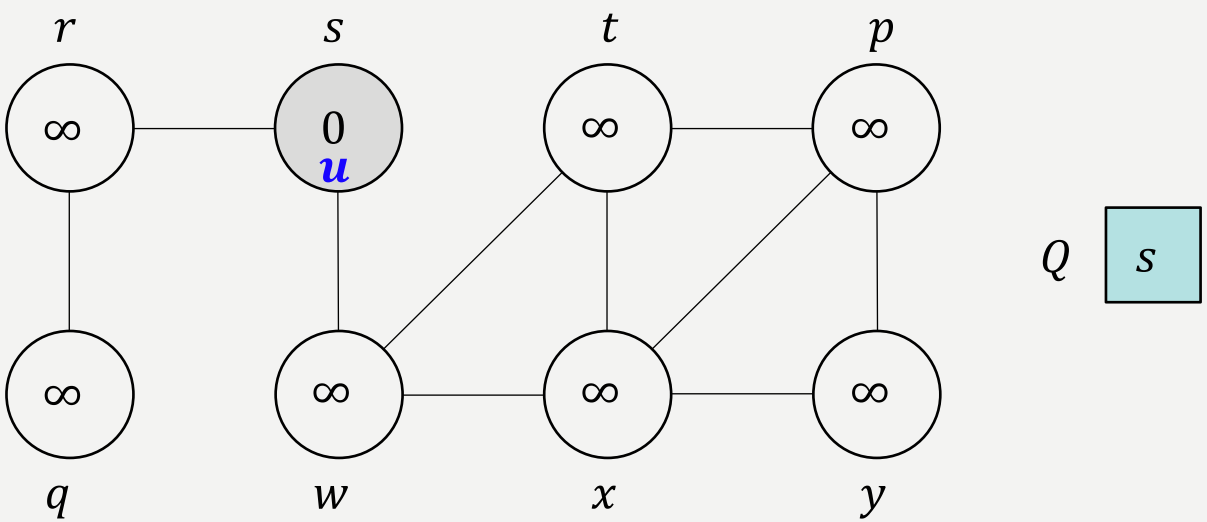
- Apply the algorithm on the graph below.



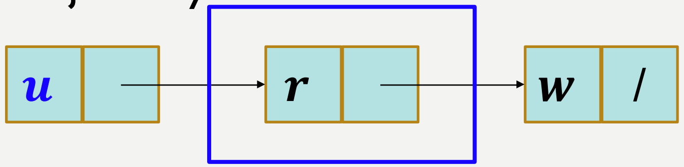

```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



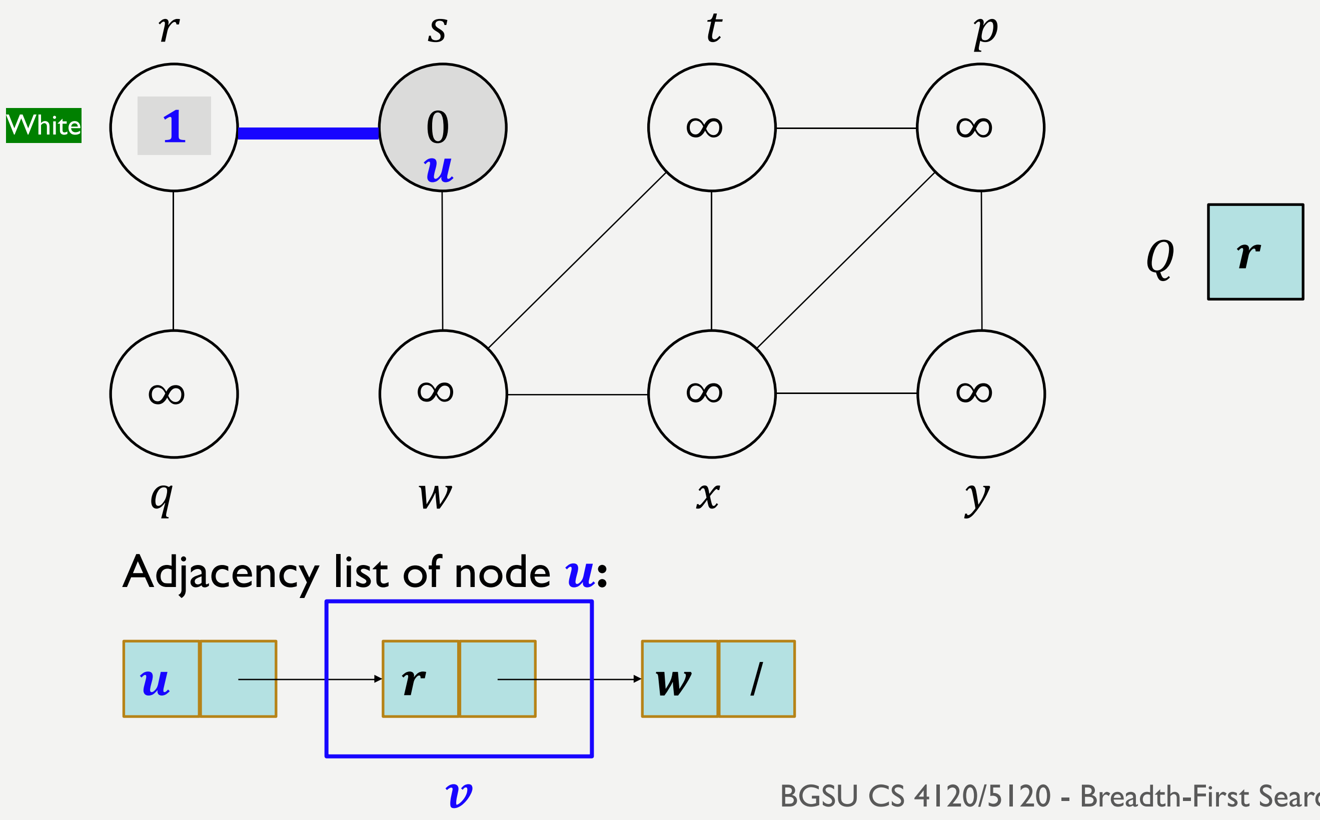
Adjacency list of node ***u***:




```
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
```

THE BFS ALGORITHM IN ACTION

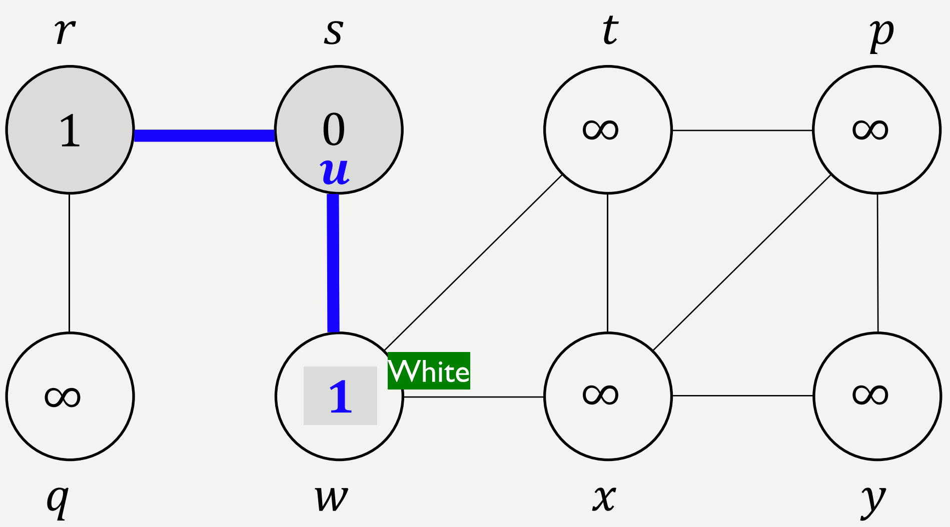
- Apply the algorithm on the graph below.

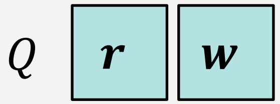


BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$\quad u.color =$ WHITE
$\quad u.d = \infty$
$\quad u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$\quad u =$ DEQUEUE (Q)
$\quad$ **for** each $v \in G.Adj[u]$
$\quad\quad$ **if** $v.color ==$ WHITE
$\quad\quad\quad v.color =$ GRAY
$\quad\quad\quad v.d = u.d + 1$
$\quad\quad\quad v.\pi = u$
$\quad\quad\quad$ ENQUEUE (Q, v)
$\quad u.color =$ BLACK

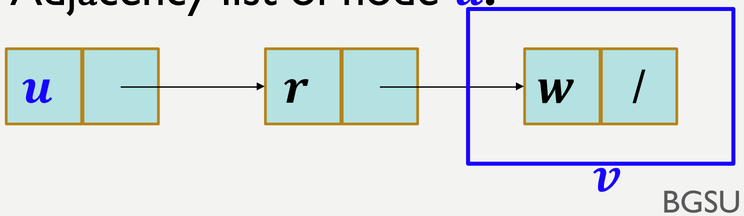
THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.





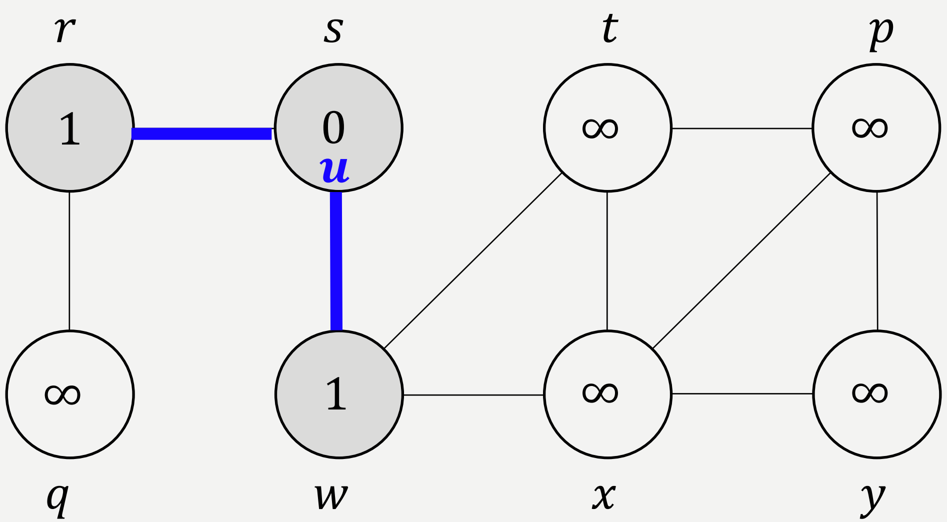
Adjacency list of node u:

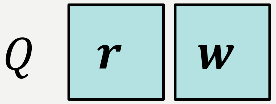



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
    u = DEQUEUE (Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π =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

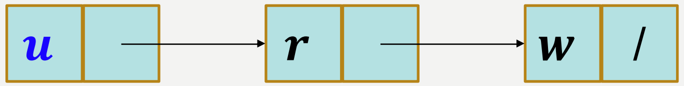
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





Adjacency list of node ***u***:

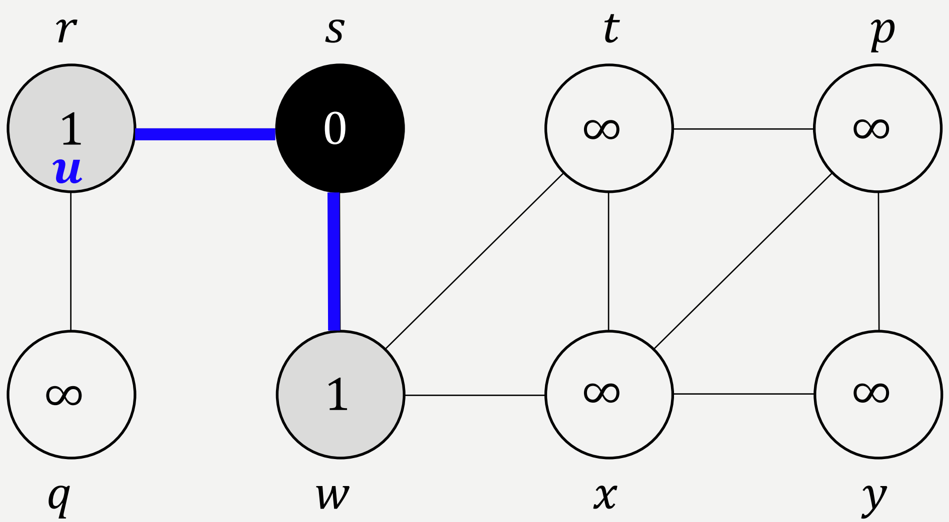


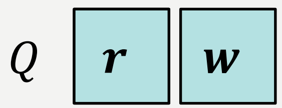
BFS (G, s)

for each vertex $u \in G.V - \{s\}$
    $u.color =$ WHITE
    $u.d = \infty$
    $u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    $u =$ DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color ==$ WHITE
            $v.color =$ GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color =$ BLACK

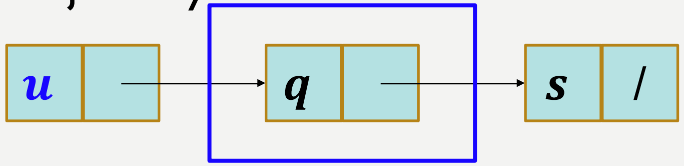
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





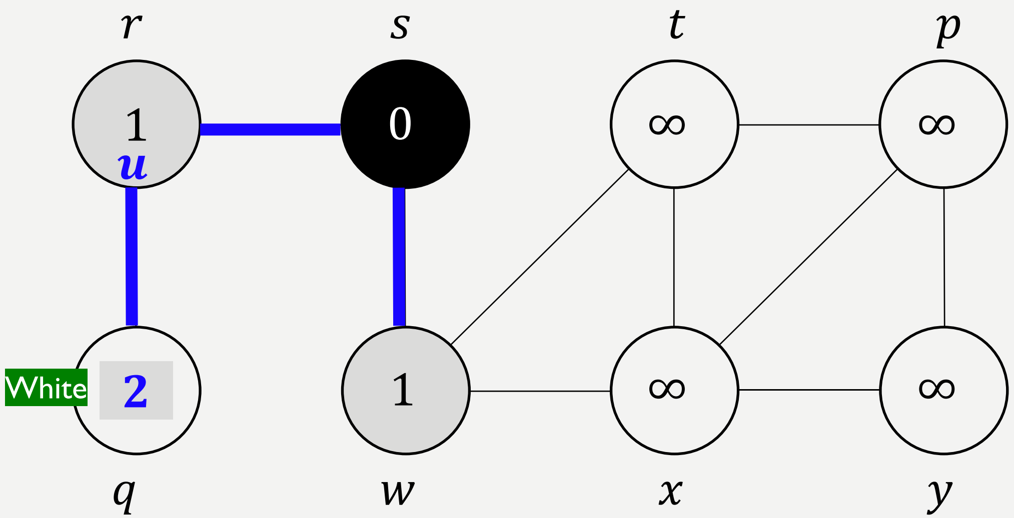
Adjacency list of node ***u***:

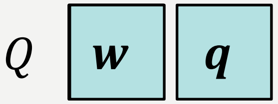



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

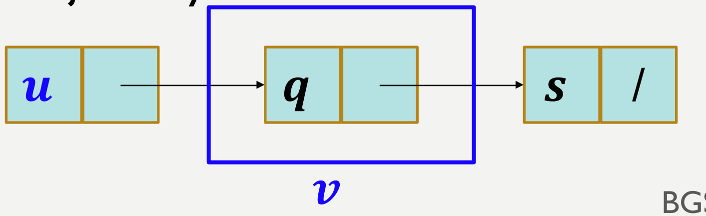
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





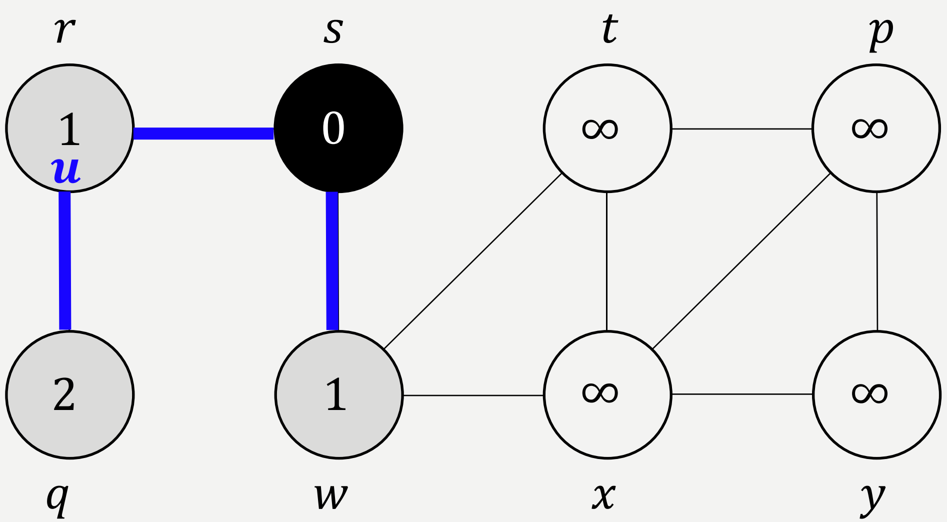
Adjacency list of node ***u***:

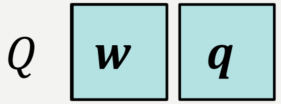


BFS (G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    u = DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
            $v.color$ = GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color$ = BLACK

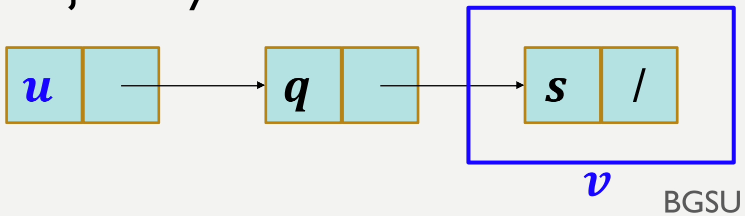
THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.





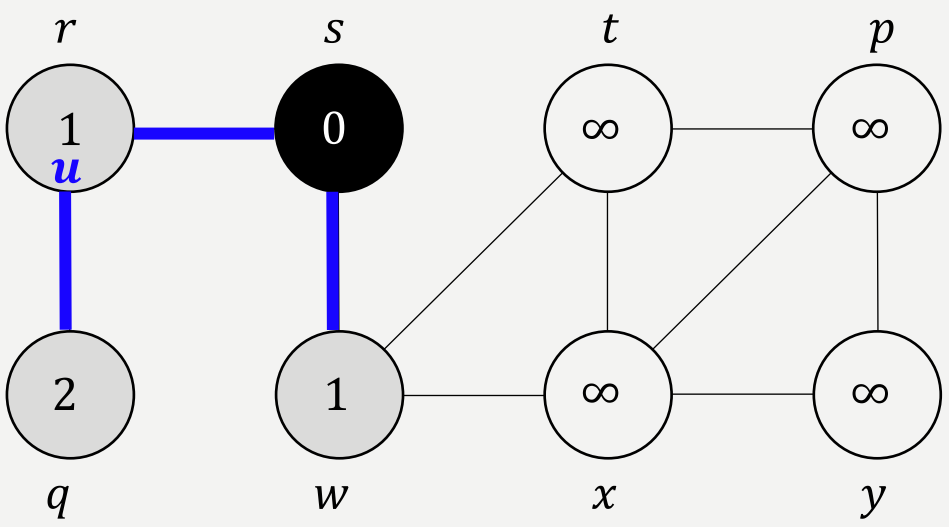
Adjacency list of node $\boldsymbol{u}$:

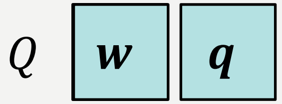



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

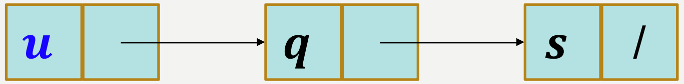
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





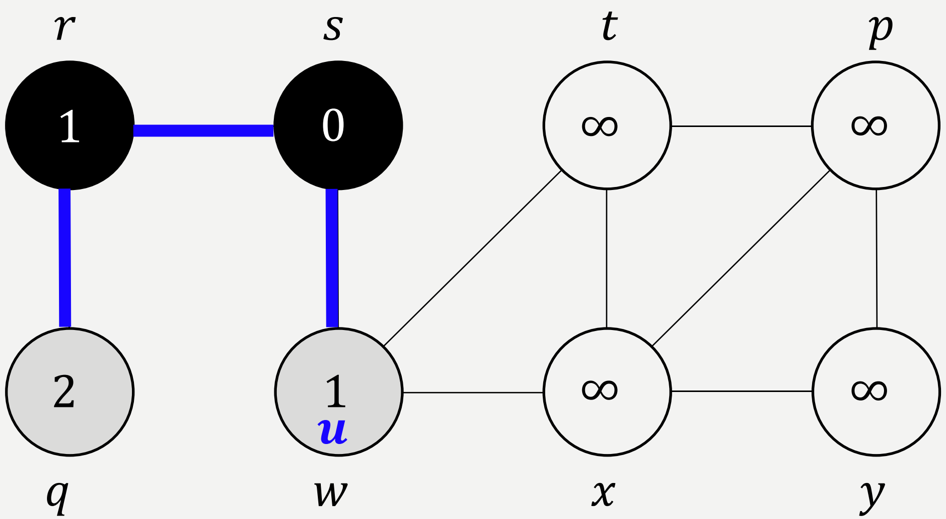
Adjacency list of node ***u***:

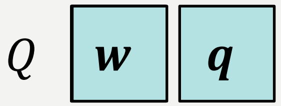



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

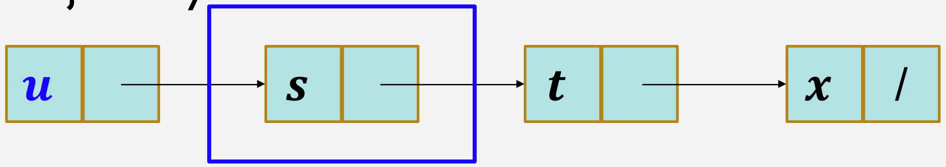
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





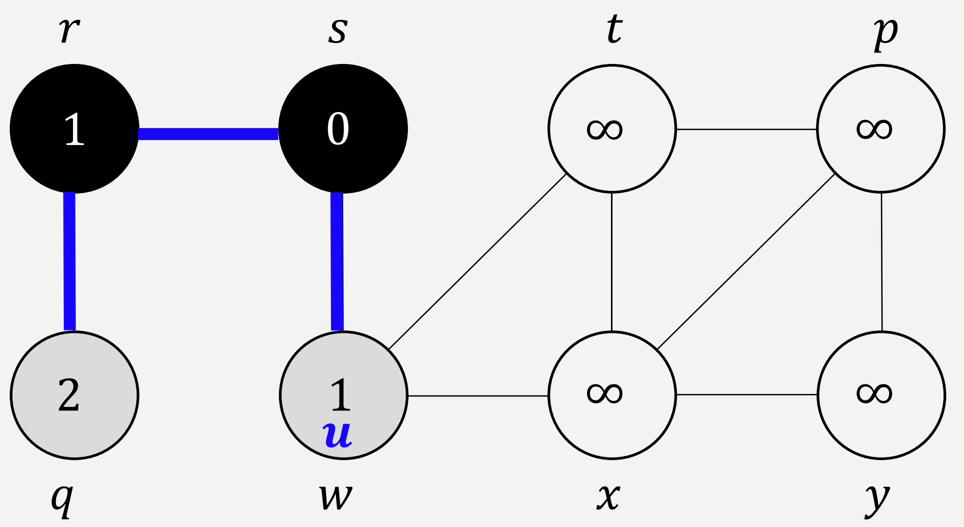
Adjacency list of node $\boldsymbol{u}$:

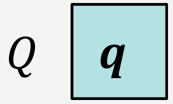


BFS (G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color =$ WHITE
    $u.d = \infty$
    $u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    $u =$ DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color ==$ WHITE
            $v.color =$ GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color =$ BLACK

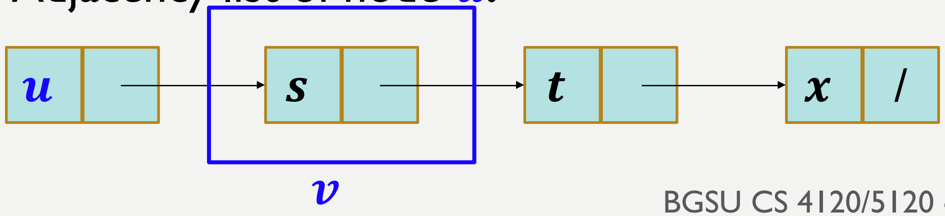
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





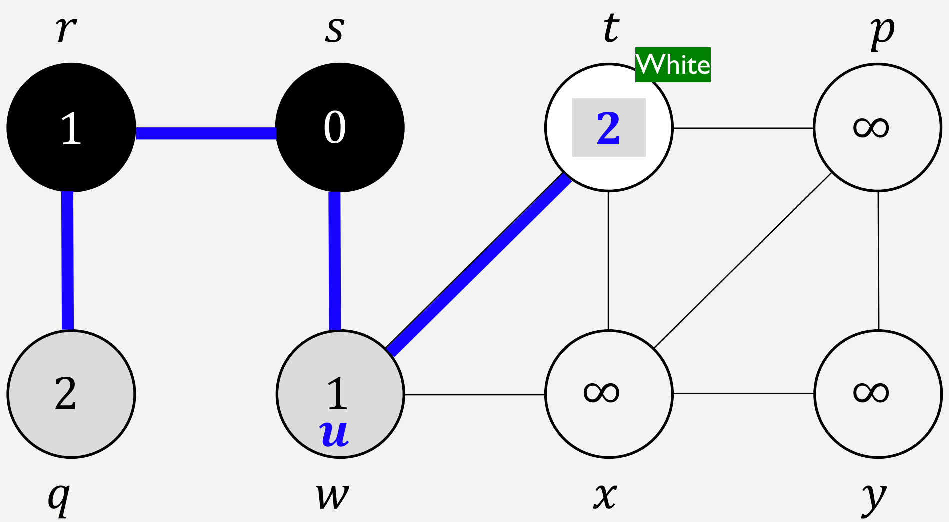
Adjacency list of node ***u***:



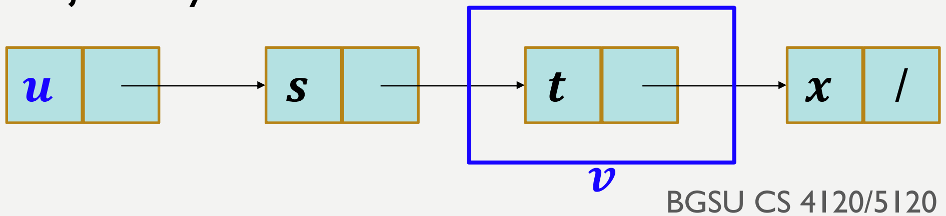

```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list of node $\boldsymbol{u}$:

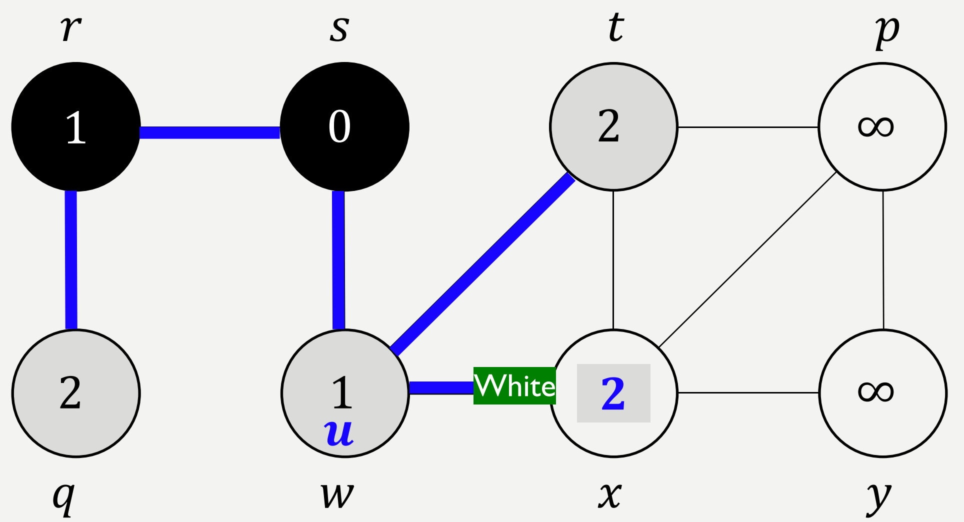


Q	$\boldsymbol{q}$	$\boldsymbol{t}$

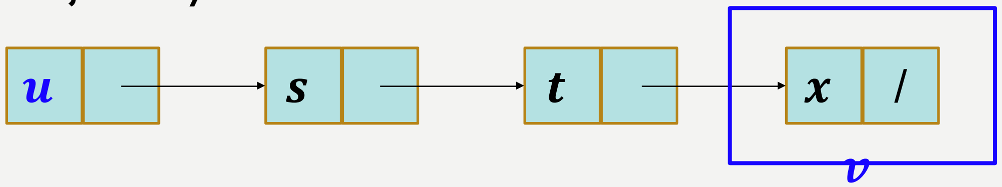
BFS (G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    u = DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
            $v.color$ = GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color$ = BLACK

THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list of node ***u***:

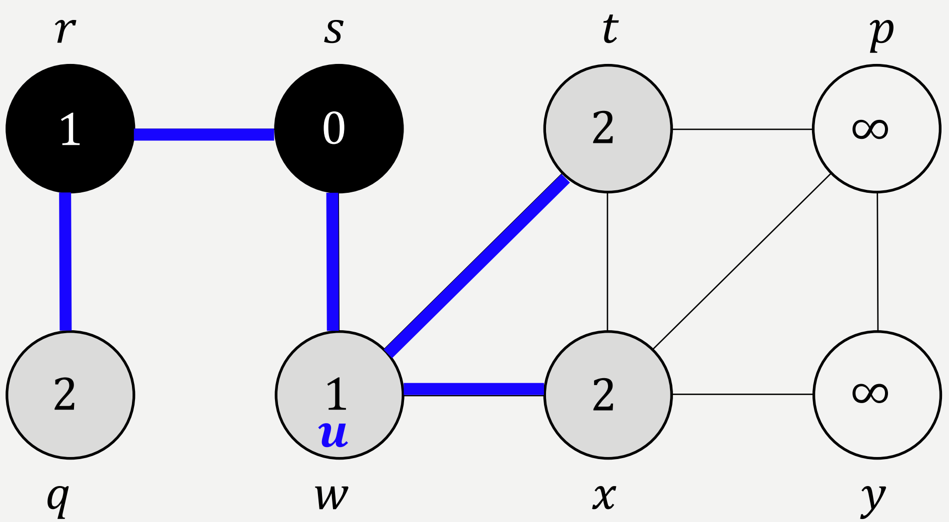


Q: ***q*** ***t*** ***x***

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$\quad u.color$ = WHITE
$\quad u.d = \infty$
$\quad u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$\quad u$ = DEQUEUE (Q)
$\quad$ **for** each $v \in G.Adj[u]$
$\quad\quad$ **if** $v.color$ == WHITE
$\quad\quad\quad v.color$ = GRAY
$\quad\quad\quad v.d = u.d + 1$
$\quad\quad\quad v.\pi = u$
$\quad\quad\quad$ ENQUEUE (Q, v)
$\quad u.color$ = BLACK

THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Q: q | t | x

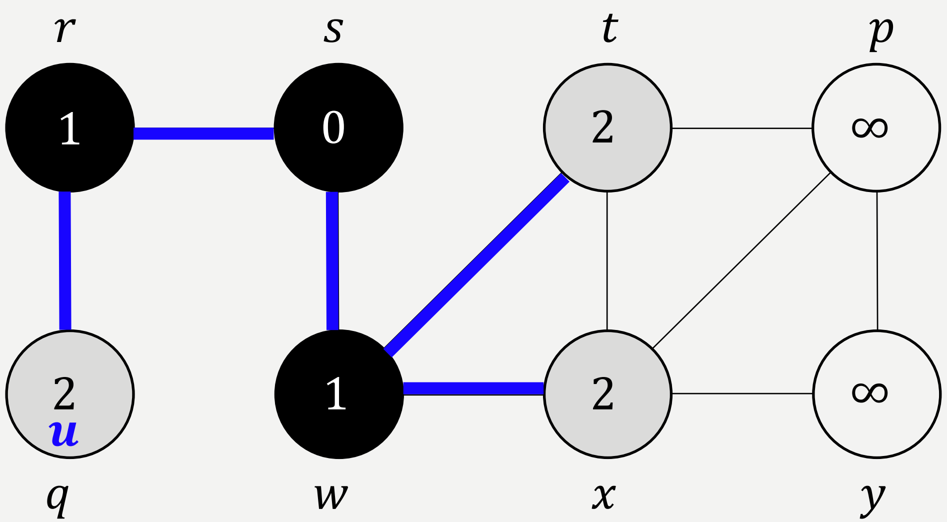
Adjacency list of node ***u***:

u → ***s*** → ***t*** → ***x*** /

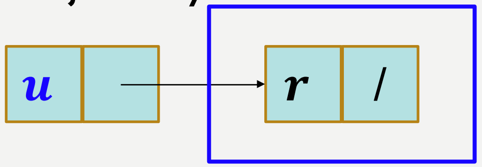
BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$\quad u.color =$ WHITE
$\quad u.d = \infty$
$\quad u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$\quad u =$ DEQUEUE (Q)
$\quad$ **for** each $v \in G.Adj[u]$
$\quad\quad$ **if** $v.color ==$ WHITE
$\quad\quad\quad v.color =$ GRAY
$\quad\quad\quad v.d = u.d + 1$
$\quad\quad\quad v.\pi = u$
$\quad\quad\quad$ ENQUEUE (Q, v)
$\quad u.color =$ BLACK

THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list of node ***u***:

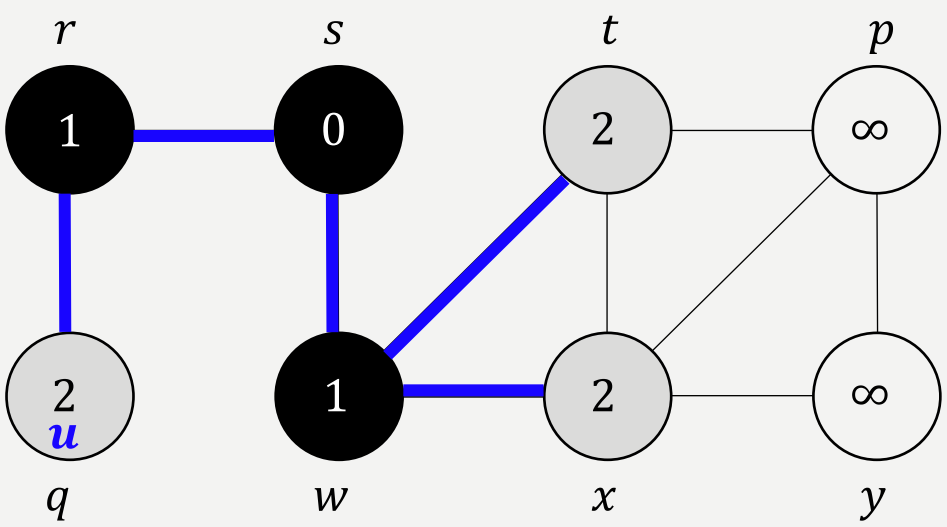


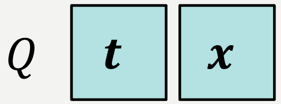
Q: ***q*** ***t*** ***x***

```
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
```

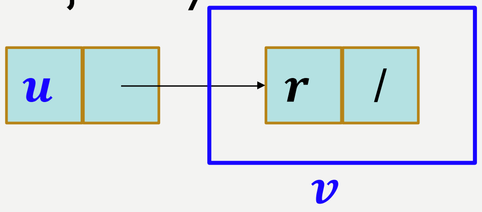
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





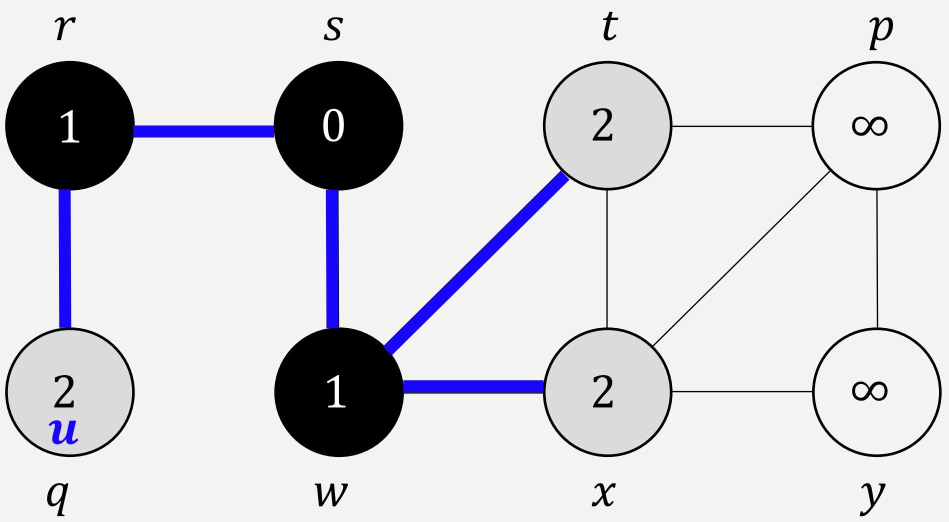
Adjacency list of node $\boldsymbol{u}$:

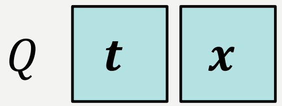


BFS (G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    u = DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
            $v.color$ = GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color$ = BLACK

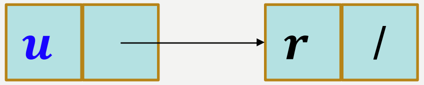
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





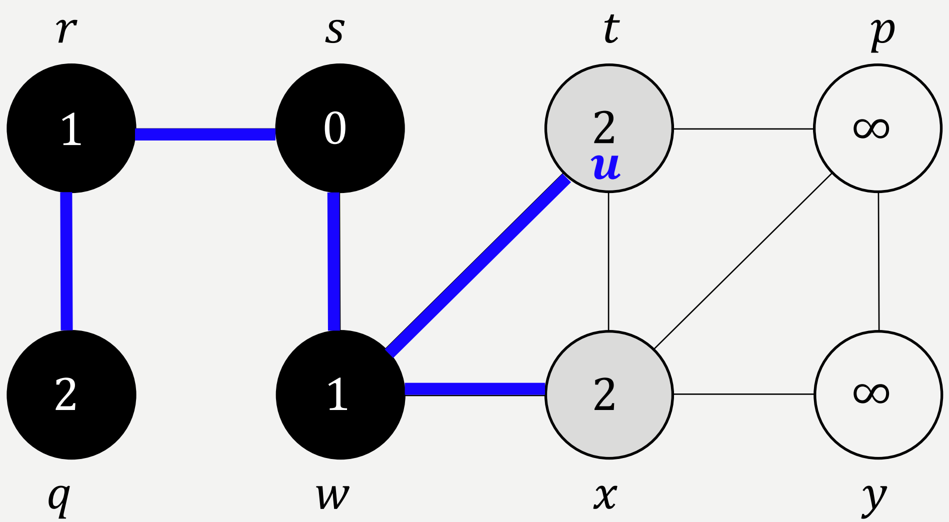
Adjacency list of node $\boldsymbol{u}$:




```
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
```

THE BFS ALGORITHM IN ACTION

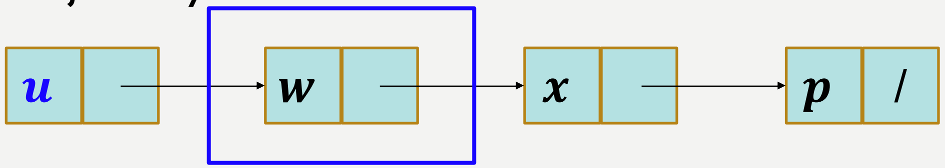
- Apply the algorithm on the graph below.



Q | t | x

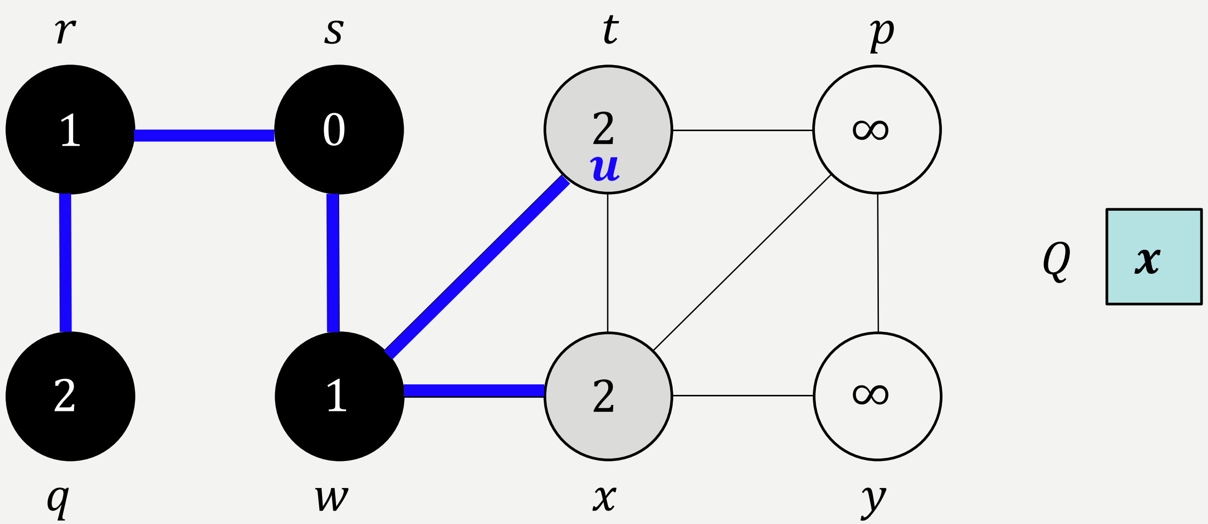
```
BFS (G, s)
for each vertex u ∈ G.V − {s}
    u.color = WHITE
    u.d = ∞
    u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
    u = DEQUEUE (Q)
    for each v ∈ G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π =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

Adjacency list of node u:

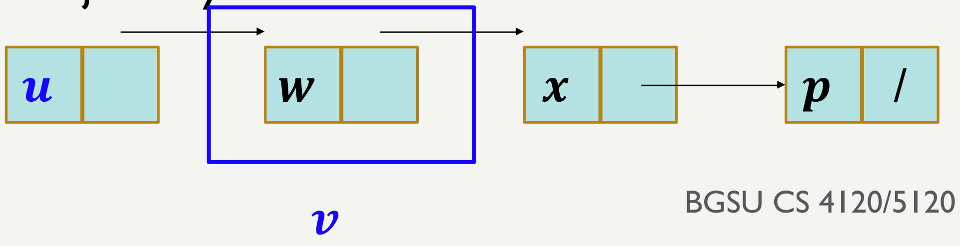


THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



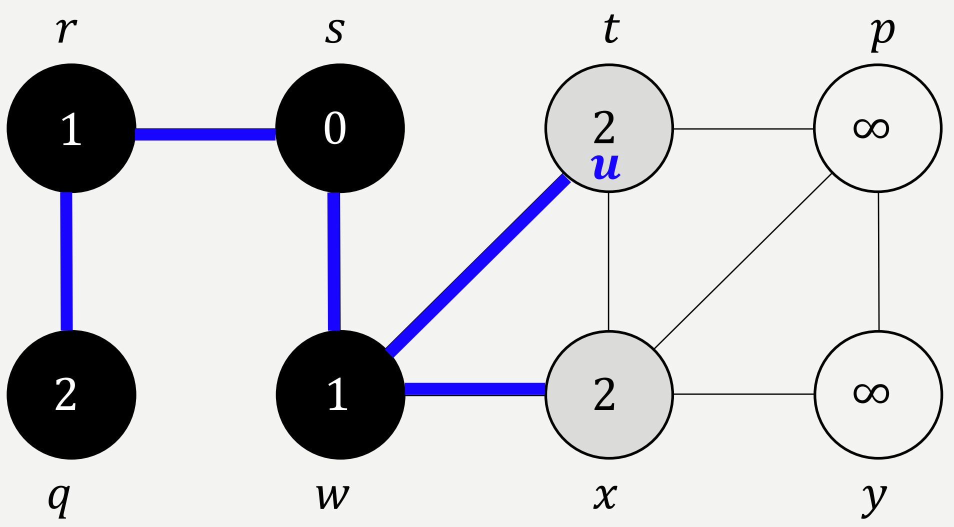
Adjacency list of node u:

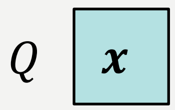



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

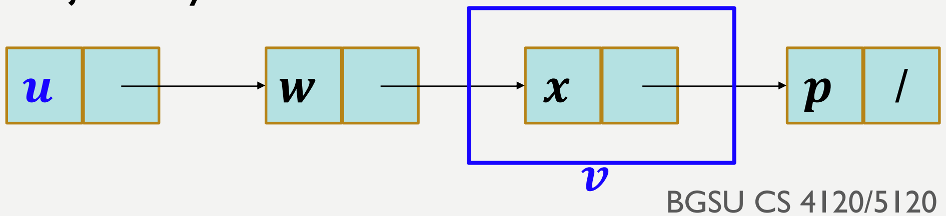
THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.





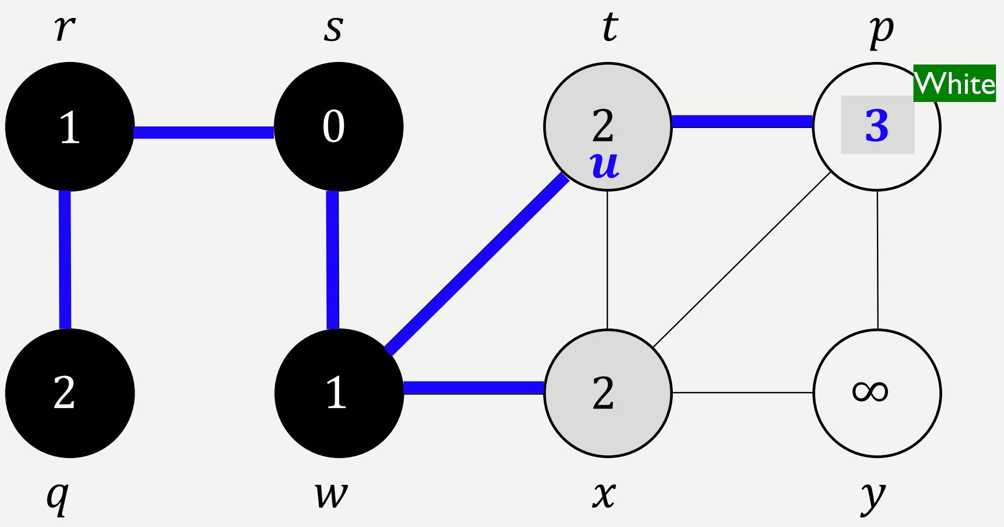
Adjacency list of node ***u***:

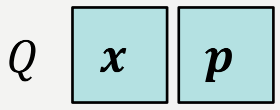



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

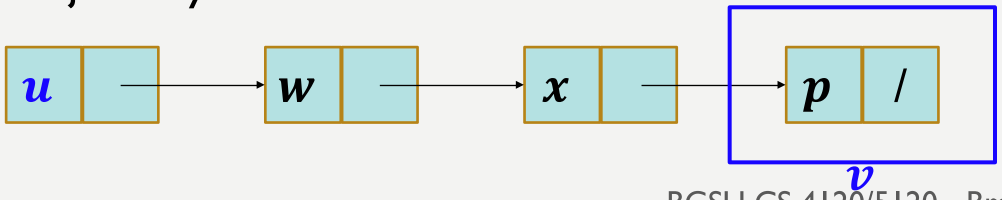
THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.





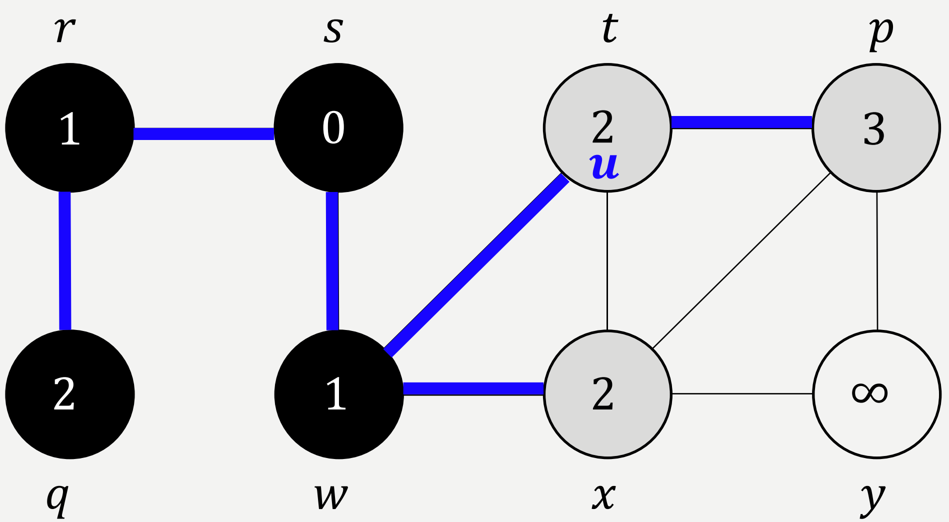
Adjacency list of node ***u***:

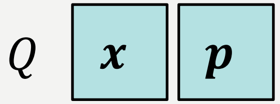



```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

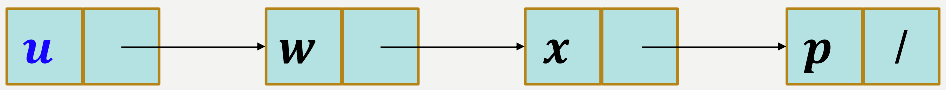
THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.





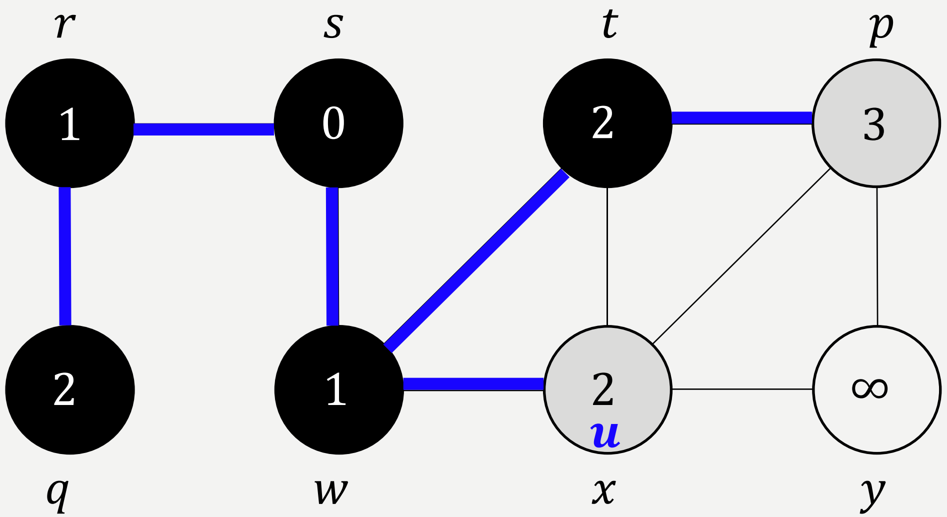
Adjacency list of node ***u***:

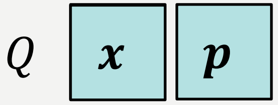


BFS (G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color =$ WHITE
    $u.d = \infty$
    $u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    $u =$ DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color ==$ WHITE
            $v.color =$ GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color =$ BLACK

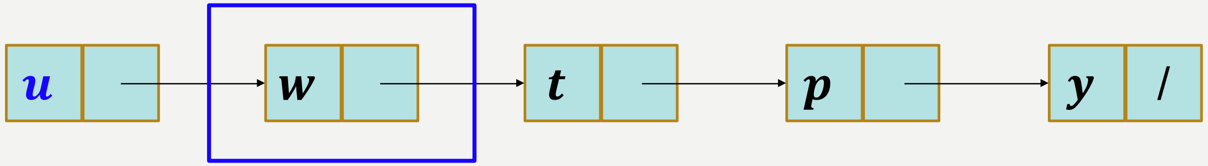
THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.





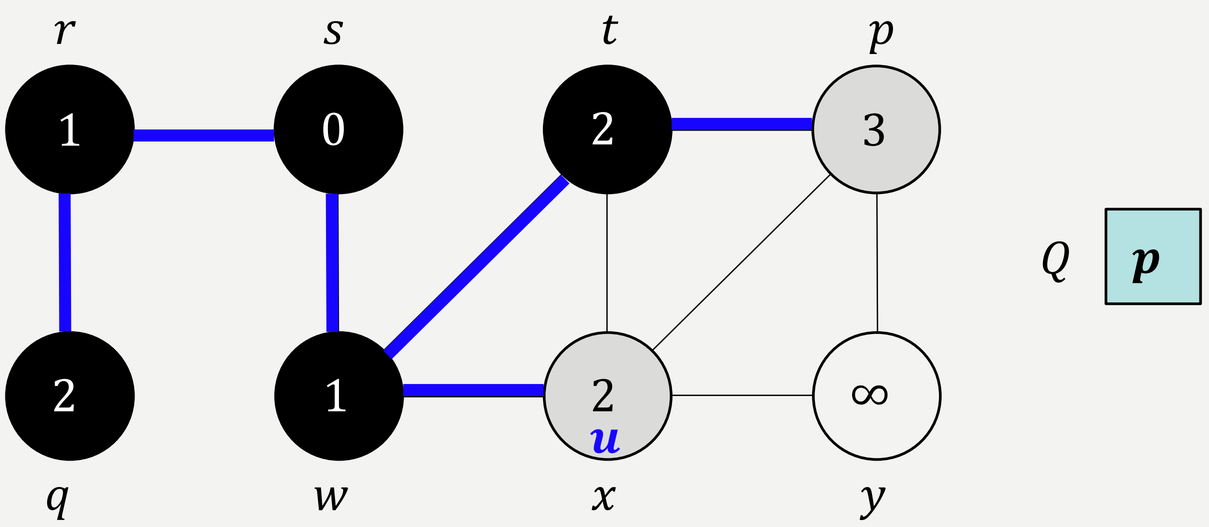
Adjacency list of node ***u***:



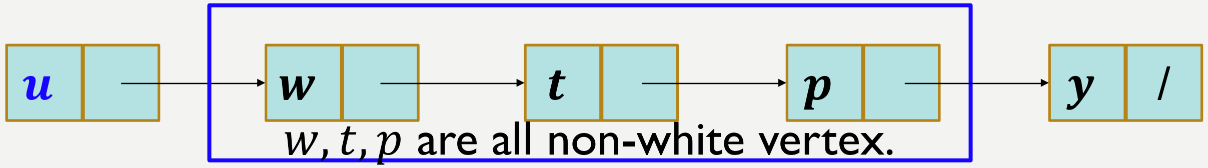

```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



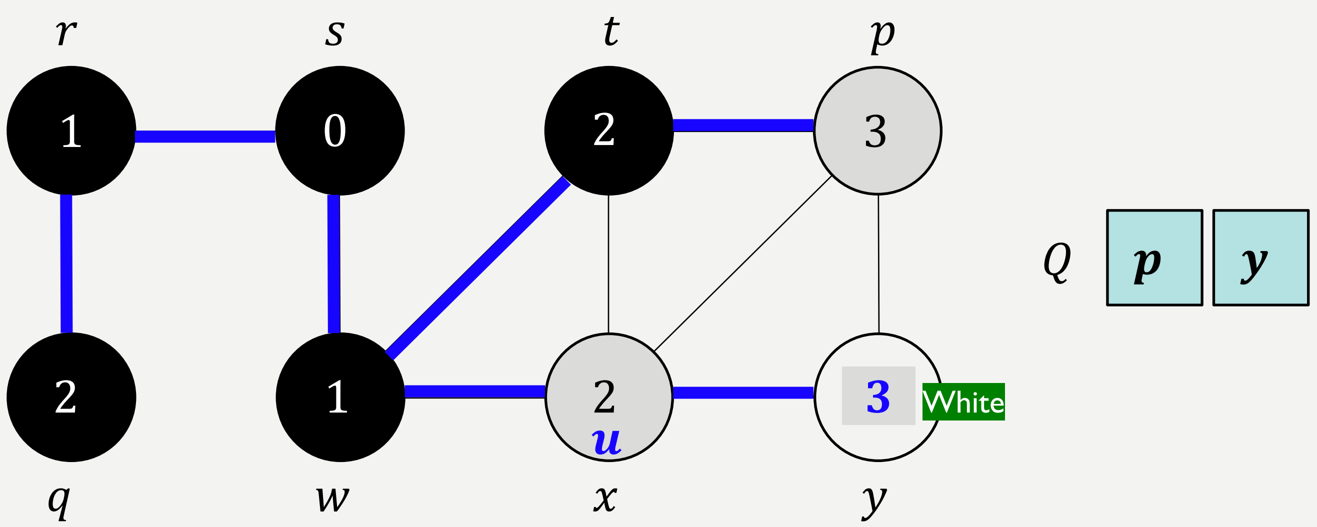
Adjacency list of node ***u***:



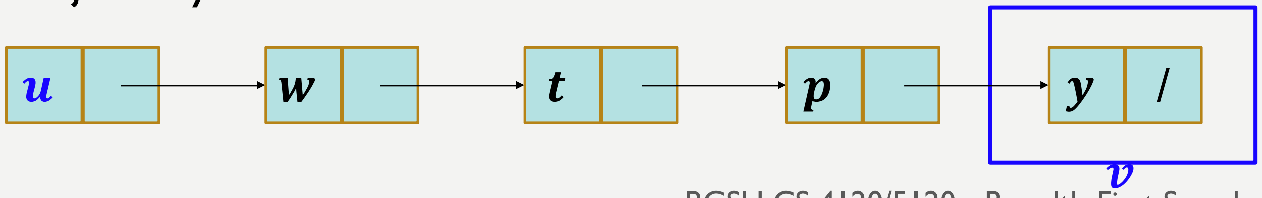

```
BFS (G, s)
for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
s.color = GRAY
s.d = 0
s.π = NIL
Q = ∅
ENQUEUE (Q, s)
while Q ≠ ∅
        u = DEQUEUE (Q)
        for each v ∈ G.Adj[u]
                if v.color == WHITE
                        v.color = GRAY
  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

THE BFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list of node ***u***:

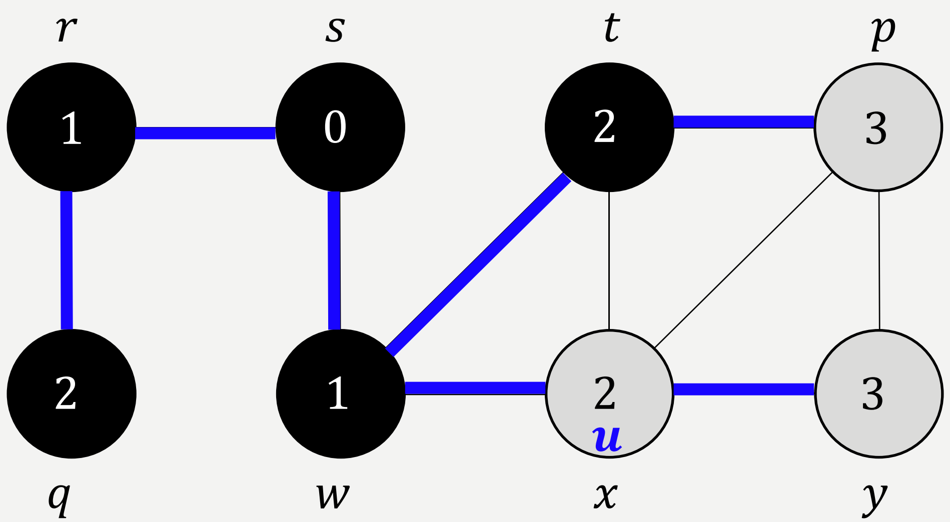


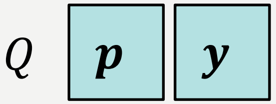
BFS (G, s)

for each vertex $u \in G.V - \{s\}$
    $u.color =$ WHITE
    $u.d = \infty$
    $u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
    $u =$ DEQUEUE (Q)
    for each $v \in G.Adj[u]$
        if $v.color ==$ WHITE
            $v.color =$ GRAY
            $v.d = u.d + 1$
            $v.\pi = u$
            ENQUEUE (Q, v)
    $u.color =$ BLACK

THE BFS ALGORITHM IN ACTION

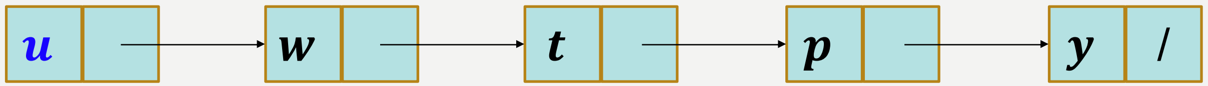
- Apply the algorithm on the graph below.






```
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
```

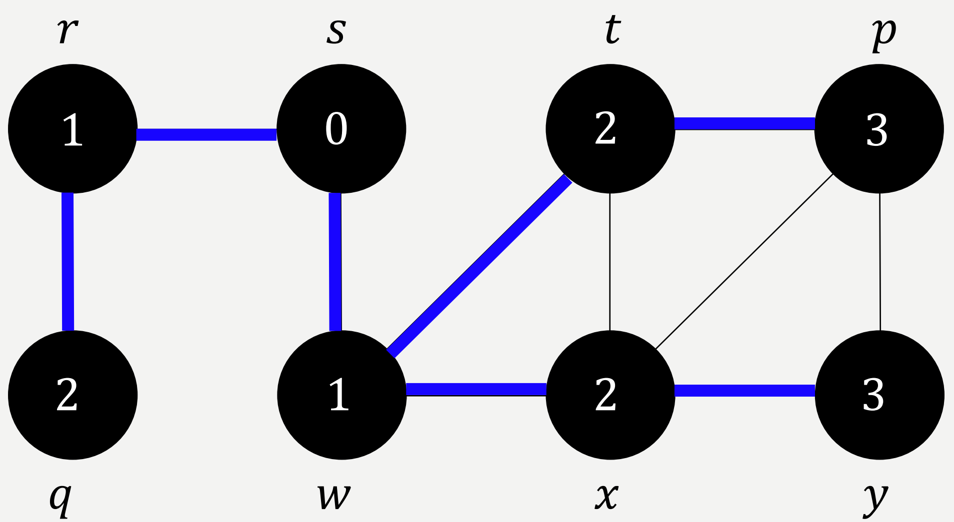
Adjacency list of node $\boldsymbol{u}$:



After checking the adjacency list of p and y, Q will be empty, and all the nodes will turn black.

THE BFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



$Q = \emptyset$

Algorithm stops.
Looking back at the procedure, each vertex is initially white, is grayed when it is ***discovered*** in the search , and is blackened when it is ***finished***.

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$\quad u.color =$ WHITE
$\quad u.d = \infty$
$\quad u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$\quad u =$ DEQUEUE (Q)
$\quad$ **for** each $v \in G.Adj[u]$
$\quad\quad$ **if** $v.color ==$ WHITE
$\quad\quad\quad v.color =$ GRAY
$\quad\quad\quad v.d = u.d + 1$
$\quad\quad\quad v.\pi = u$
$\quad\quad\quad$ ENQUEUE (Q, v)
$\quad u.color =$ BLACK

THE BFS ALGORITHM
RUNNING TIME - INIT

- Initialization
  - The algorithm whitens $\boldsymbol{|V|-1}$ vertices initially, which takes $\boldsymbol{\Theta(|V|)}$ time.
  - After initialization it never whitens any vertex.

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$u.color =$ WHITE
$u.d = \infty$
$u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$u =$ DEQUEUE (Q)
for each $v \in G.Adj[u]$
if $v.color ==$ WHITE
$v.color =$ GRAY
$v.d = u.d + 1$
$v.\pi = u$
ENQUEUE (Q, v)
$u.color =$ BLACK

THE BFS ALGORITHM RUNNING TIME - QUEUE

- Queue operation
  - Each node is enqueued at most ___one___ time(s), and dequeued at most ___one___ time(s).
  - The time complexity of one enqueue operation is ___$\Theta(1)$___, and the time complexity of dequeue operation is ___$\Theta(1)$___.
  - There are ___$\Theta(|V|)$___ vertices.
  - Therefore, total time devoted to queue operations is ___$\Theta(|V|)$___.

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$u.color =$ WHITE
$u.d = \infty$
$u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$u =$ DEQUEUE (Q)
for each $v \in G.Adj[u]$
if $v.color ==$ WHITE
$v.color =$ GRAY
$v.d = u.d + 1$
$v.\pi = u$
ENQUEUE (Q, v)
$u.color =$ BLACK

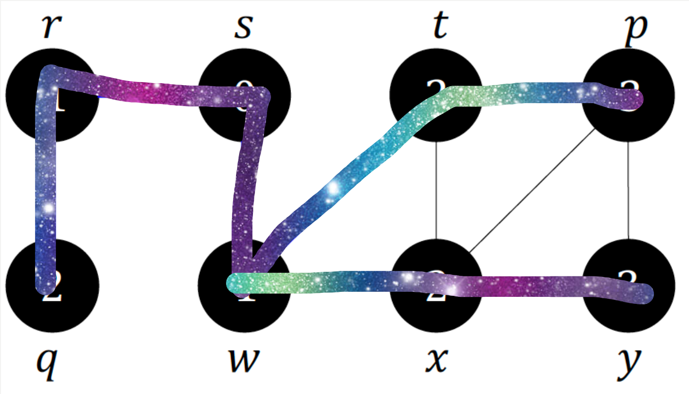
THE BFS ALGORITHM
RUNNING TIME

- Scanning adjacency list
  - The Algorithm scans each adjacency list at most <u>one</u> time(s).
  - The sum of lengths of all the adjacency lists is $\boldsymbol{\Theta(|E|)}$, the total time spent in scanning the adjacency lists is $\boldsymbol{\Theta(|E|)}$.
- The total running time of BFS is $\boldsymbol{\Theta(|V| + |E|)}$.

BFS (G, s)
for each vertex $u \in G.V - \{s\}$
$u.color =$ WHITE
$u.d = \infty$
$u.\pi =$ NIL
$s.color =$ GRAY
$s.d = 0$
$s.\pi =$ NIL
$\boldsymbol{Q} = \emptyset$
ENQUEUE (Q, s)
while $Q \neq \emptyset$
$u =$ DEQUEUE (Q)
for each $v \in G.Adj[u]$
if $v.color ==$ WHITE
$v.color =$ GRAY
$v.d = u.d + 1$
$v.\pi = u$
ENQUEUE (Q, v)
$u.color =$ BLACK

THE BFS ALGORITHM
A BRIEF SUMMARY

- Utilizes the ***adjacency list*** data structure to represent a graph.
- The algorithm
  - initially ***white***ns the vertices
  - ***gray***s a vertex when it's first discovered
  - ***blacken***s a vertex when it is exhausted
- When finishing the algorithm, we have traversed all the vertices while recording the path that we took when we discover each vertex for the first time.



THE BREADTH-FIRST TREES

- The BFS procedure builds a ***breadth-first tree*** as it searches the graph.
  - The tree corresponds to the π attributes.
- Formally, for a graph $G = (V, E)$ with source s, we define the ***predecessor subgraph*** of G as $G_\pi = (V_\pi, E_\pi)$, where

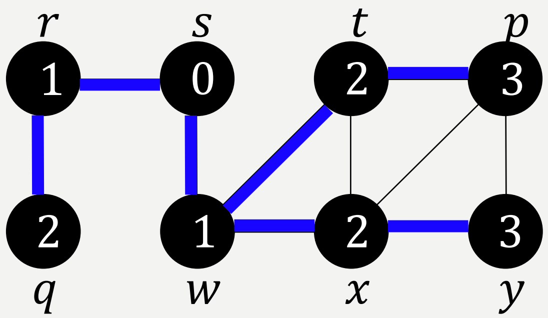
$$V_\pi = \{v \in V : v.\pi \neq NIL\} \cup \{s\}$$

Source s is included

and

$$E_\pi = \{(v.\pi, v) : v \in V_\pi - \{s\}\}$$

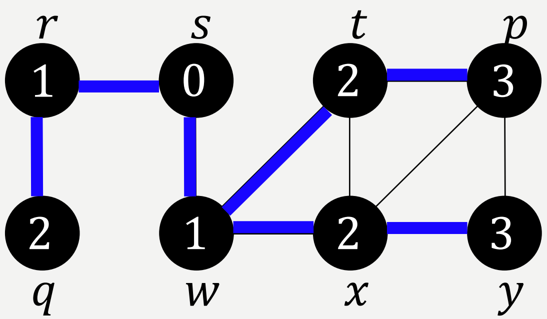
Source s is represented by $v.\pi$

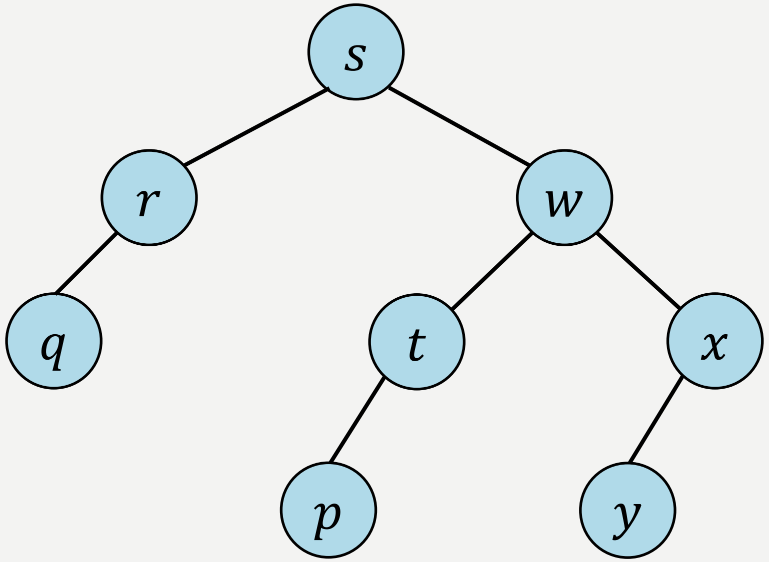


BREADTH-FIRST TREES LEMMA 22.6

- When applied to a directed or undirected graph $G = (V, E)$, procedure BFS constructs π so that the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ is a breadth-first tree.

- What is the corresponding tree to the graph below?





SHORTEST PATHS IN BFS

- Given a graph $G = (V, E)$ from a given source vertex $s \in V$.
- Define the ***shortest-path distance*** $\boldsymbol{\delta(s, v)}$ from s to v as the ***minimum*** number of edges in any path from vertex s to vertex v; if there is no path from s to v, then $\delta\ (s, v) = \infty$.
- A **path of length** $\boldsymbol{\delta(s, v)}$ from s to v is called a ***shortest path***.

BFS AND SHORTEST PATHS THEOREM 22.5

- Let $G = (V, E)$ be a ***directed or undirected*** graph, and suppose that BFS is run on G from a given source vertex $s \in V$. Then during its execution, BFS discovers every vertex $v \in V$ that is reachable from the source s, and **upon termination, $\boldsymbol{v.d = \delta(s, v)}$ for all $\boldsymbol{v \in V}$**. Moreover, for any vertex $v \neq s$ that is reachable from s, one of the shortest paths from s to v is a shortest path from s to $v.\pi$ followed by the edge $(v.\pi, v)$.

- Denote the length of an edge (u, v) by $e(u, v)$, then the above theorem can be formulated as a recurrence

$$\delta(s, v) = \underline{\qquad\qquad\qquad} + \underline{\qquad\qquad\qquad}$$

NEXT UP
DEPTH-FIRST SEARCH

REFERENCE

- https://www.netclipart.com/isee/hRwxRh_kids-clipart-nurse-cute-female-doctor-cartoon/
- https://listposts.com/lera-kiryakova-celebrities-cartoon-characters/
- https://www.computerhope.com/people/david_huffman.htm