Graph Distances: Theory & Computation

Lecture VII for course in Data Structures & Algorithms for AI

Lectures by Hannah Santa Cruz Baur.

Based on *Introduction to Algorithms*, Cormen et al. and material by Rafaella Mulas.

Homework (due with Assignment 3)

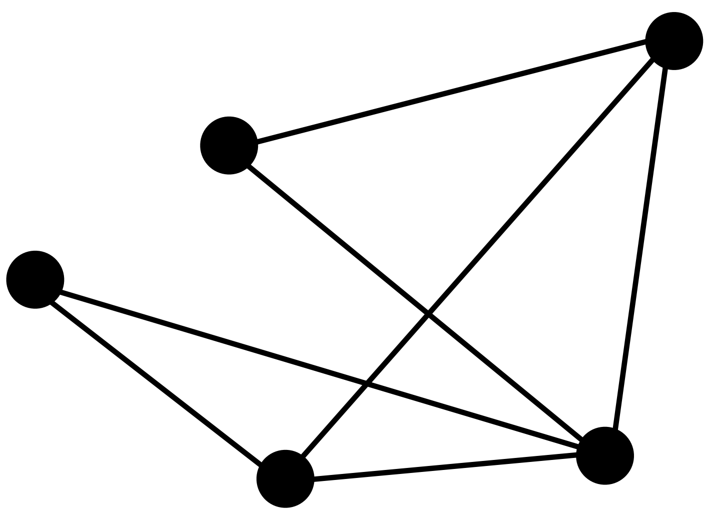
- The square of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ if and only if G contains a walk with two edges between u and v.

  Describe an efficient algorithm for computing G^2 from the adjacency matrix of G, and analyze the running time.

- Most graph algorithms that take an adjacency-matrix representation as input require $\Omega(V^2)$ time, but there are some exceptions. Show how to determine whether a directed graph contains a universal sink - a vertex with in-degree $|V| - 1$ and out-degree 0 in $O(V)$ time, given the adjacency matrix.

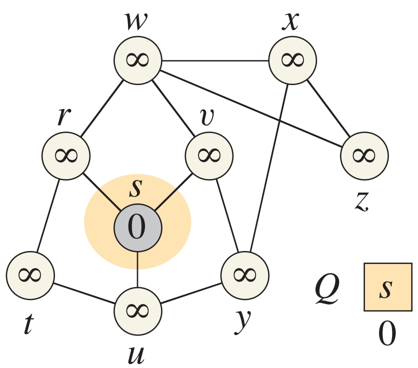
Homework (due with Assignment 3)

- Using matrix multiplication, compute the number of walks on 3 edges between al pairs of vertices, for the graph bellow. Your answer should be a square matrix on the number of vertices.



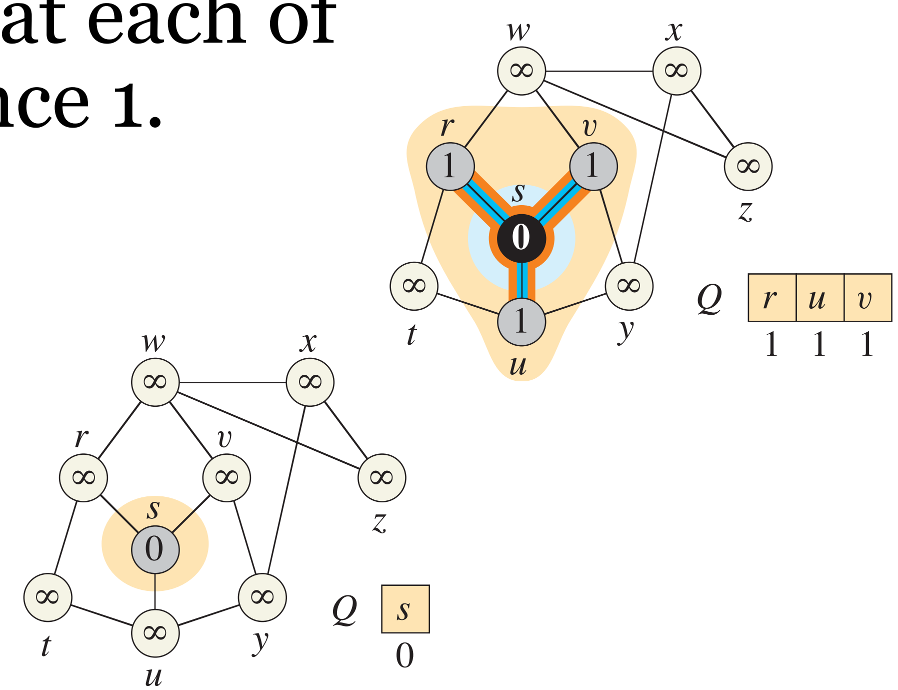
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.



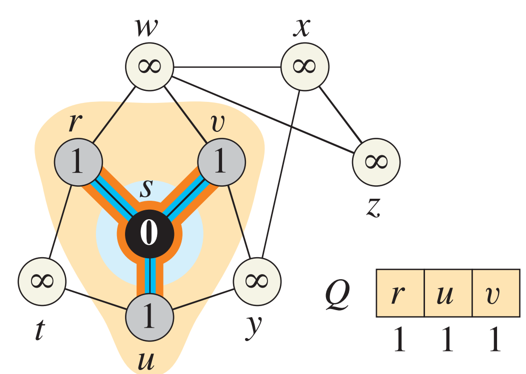
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- We begin by looking at each of it's neighbors- distance 1.

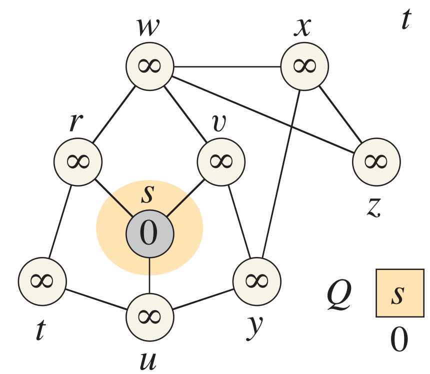


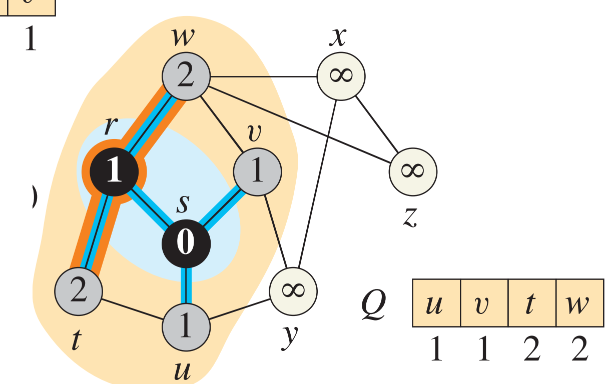
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- We begin by looking at each of it’s neighbors- distance 1.



- Next, we expand the frontier - one neighbor at at time.

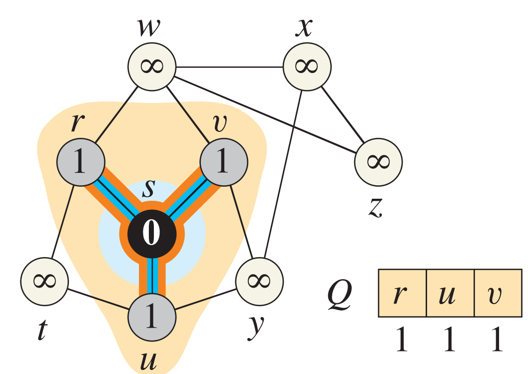




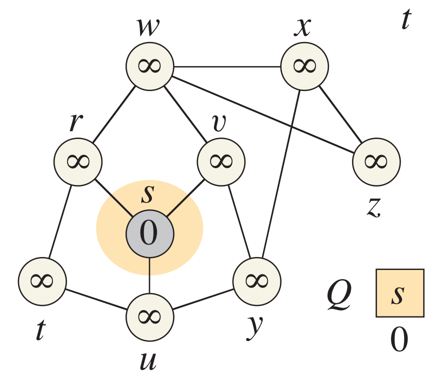
Now discovering the neighbors of neighbors.

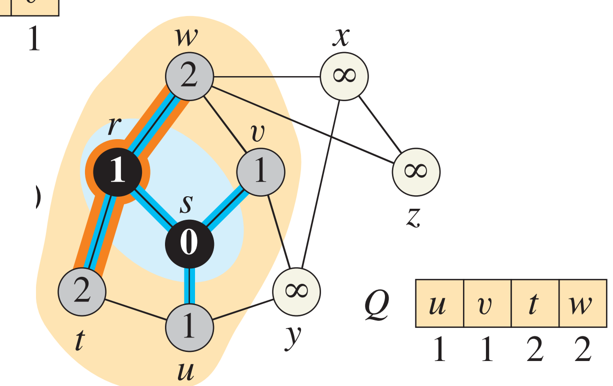
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- We begin by looking at each of it's neighbors- distance 1.

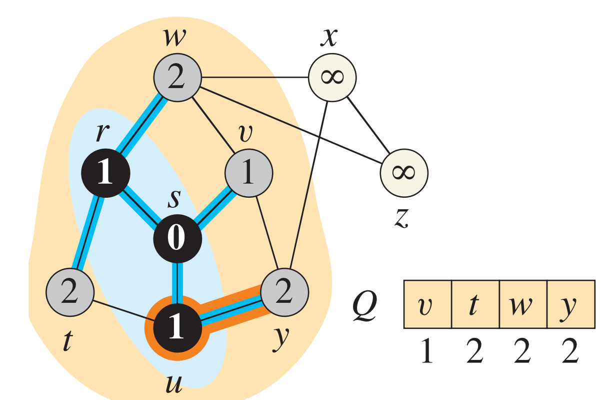


- Next, we expand the frontier - one neighbor at at time.



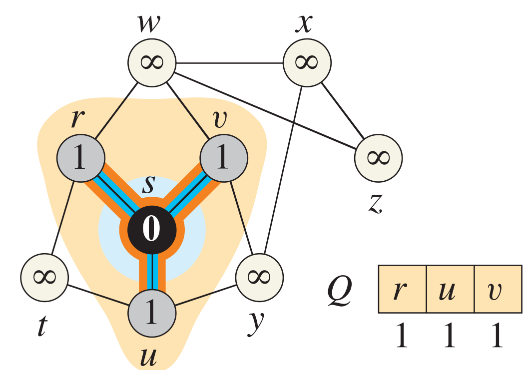


Now discovering the neighbors of neighbors.

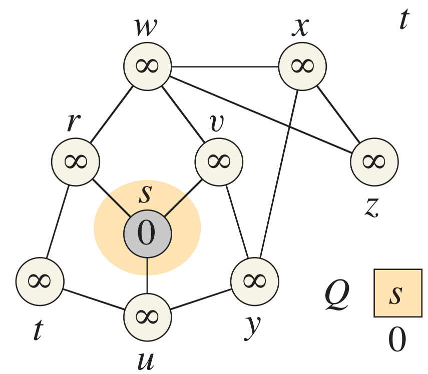


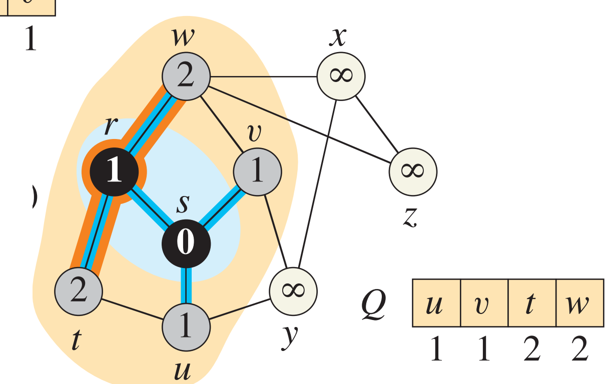
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- We begin by looking at each of it's neighbors- distance 1.

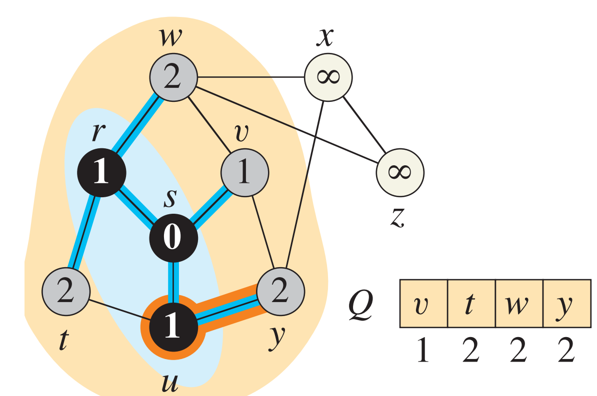


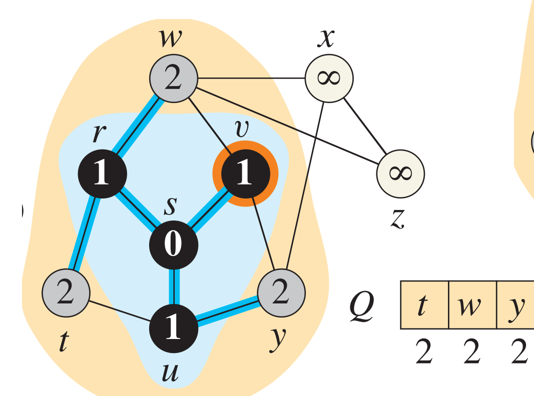
- Next, we expand the frontier - one neighbor at at time.





Now discovering the neighbors of neighbors.

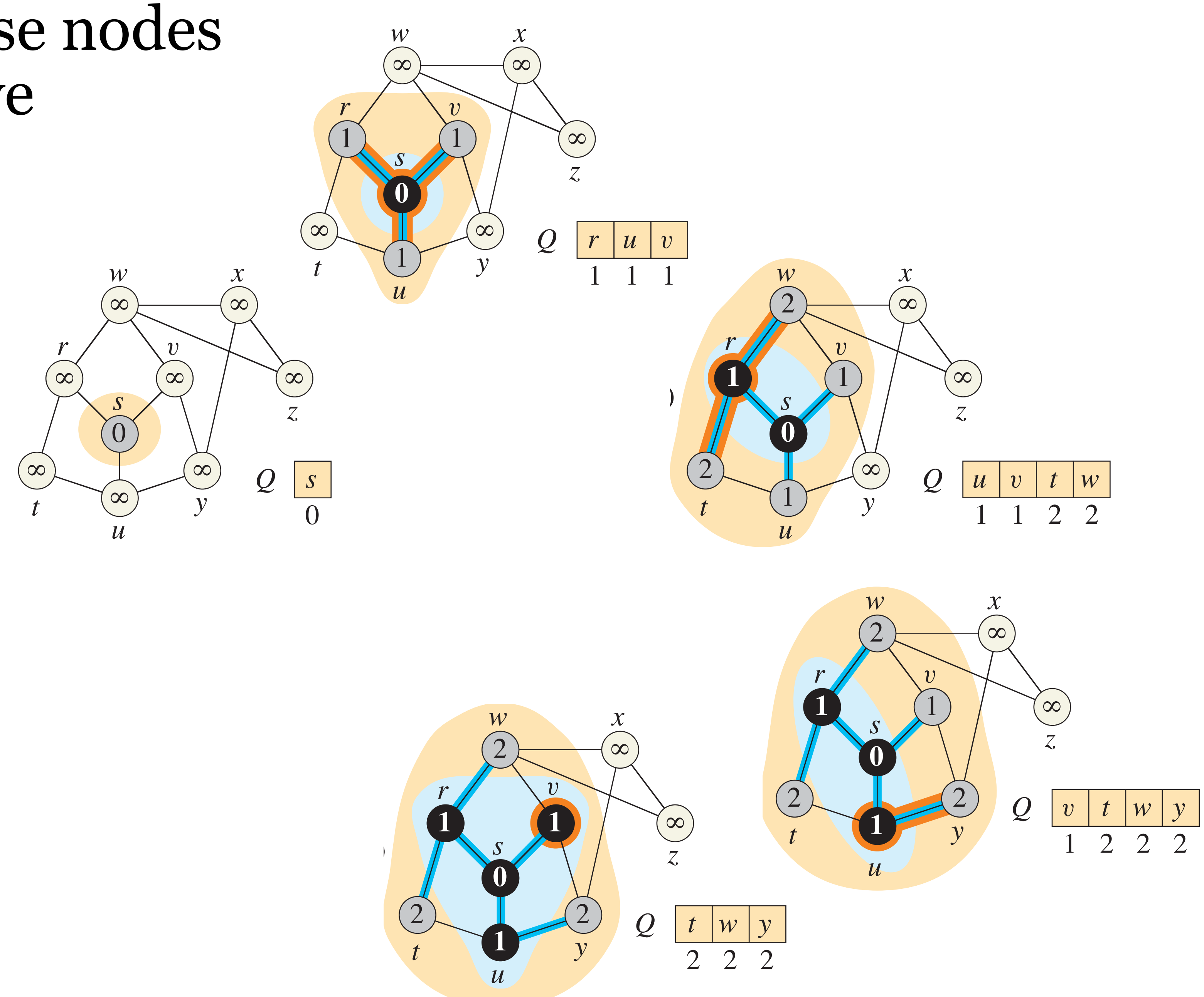




Breadth First Search

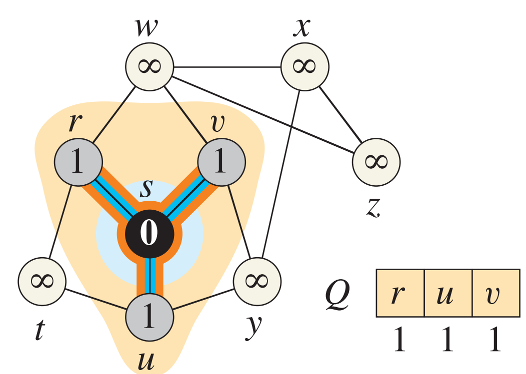
- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- In blue, we track those nodes whose neighbors have been discovered.

  In tan we track discovered nodes with yet unexplored neighbors.

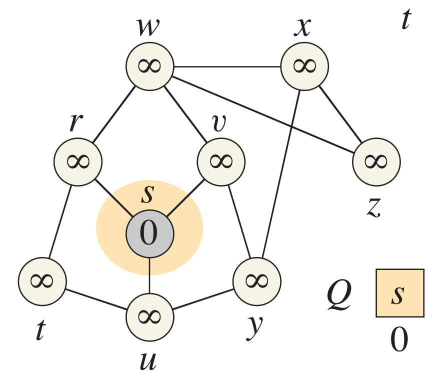


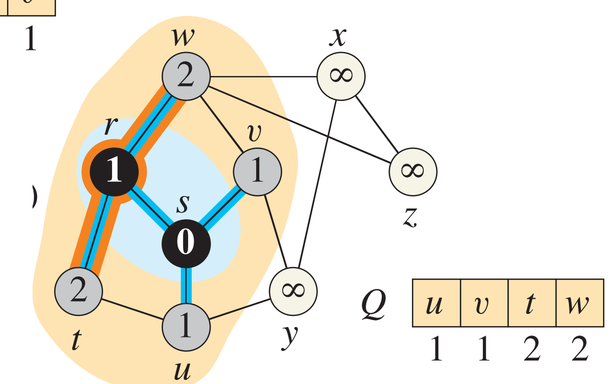
Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- In blue, we track those nodes whose neighbors have been discovered.

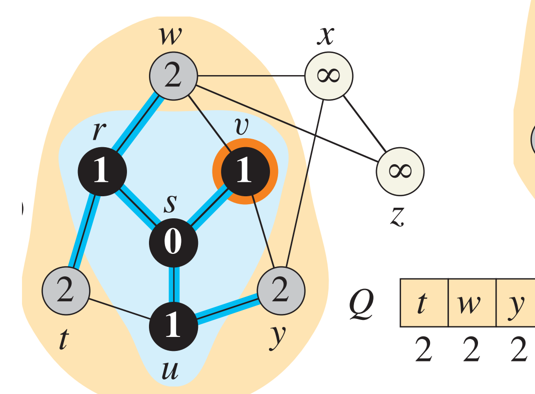


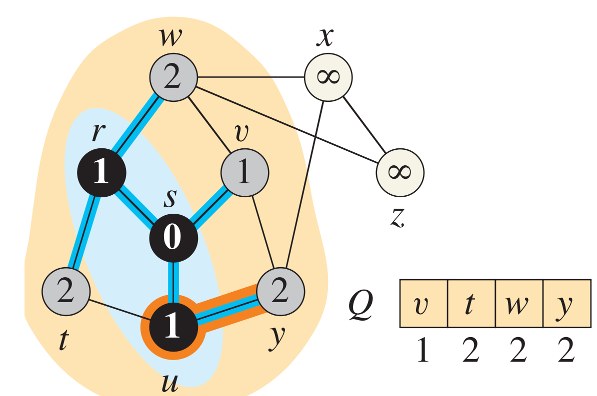
In tan we track discovered nodes with yet unexplored neighbors.





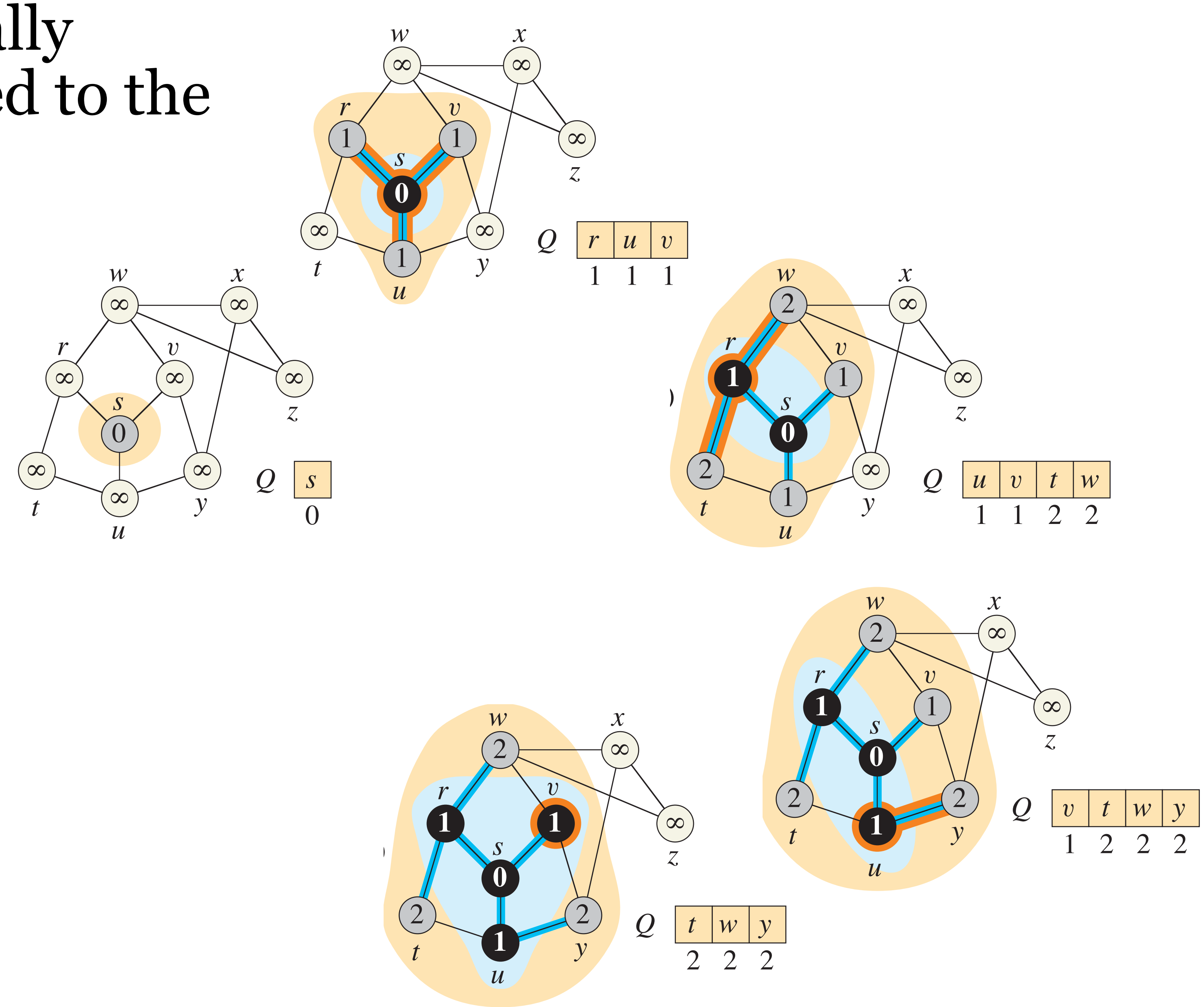
Nodes with yet unexplored neighbors, are kept track of with a queue, Q.





Breadth First Search

- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- When a node is initially discovered, it is added to the queue.

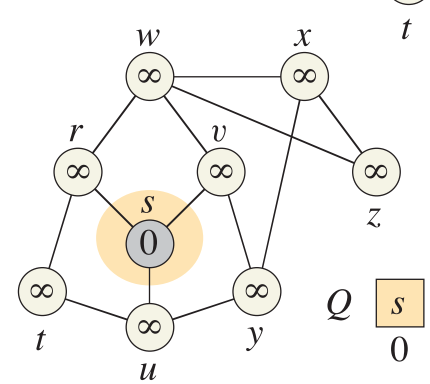


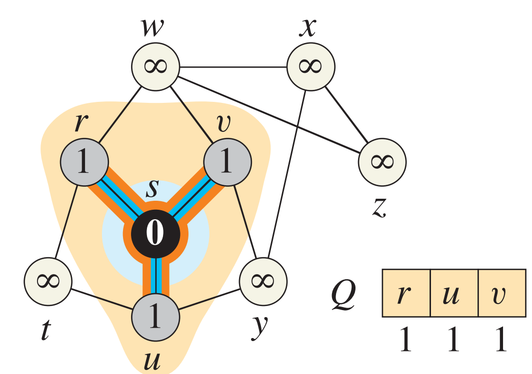
Breadth First Search

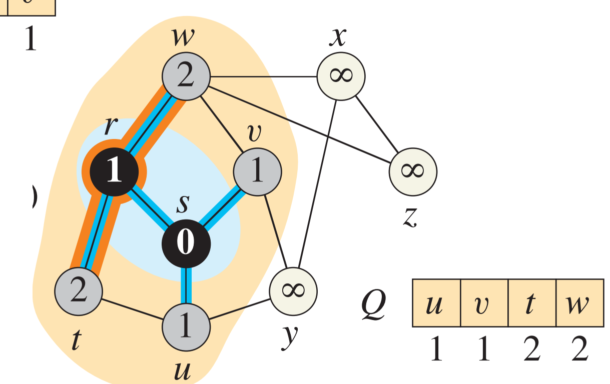
- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- When a node is initially discovered, it is added to the queue.

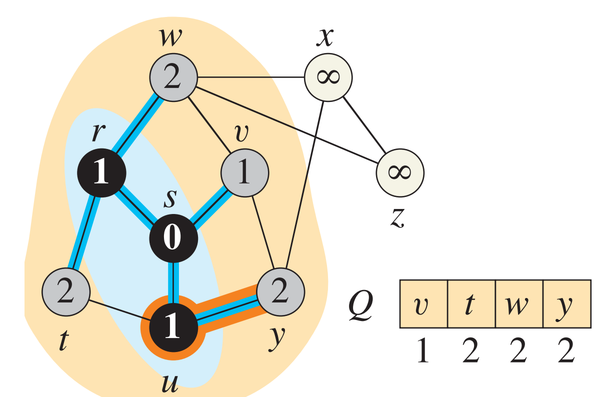
  It's distance to the source node is recorded.

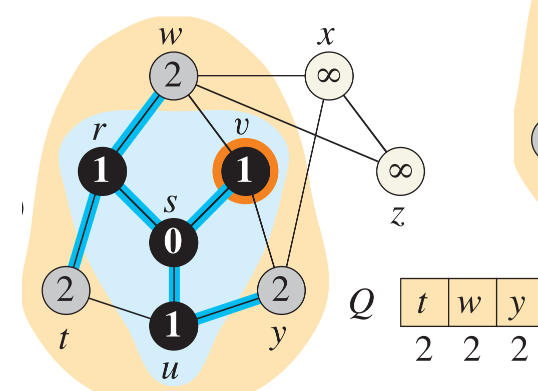
  And the node is colored, grey.







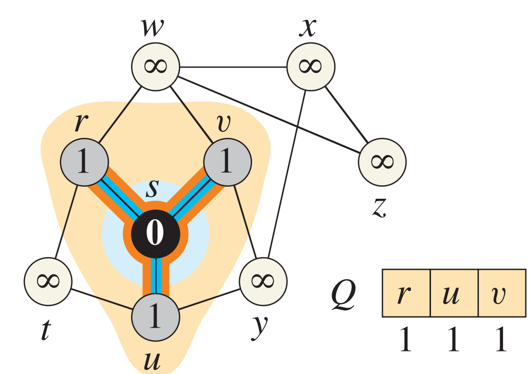




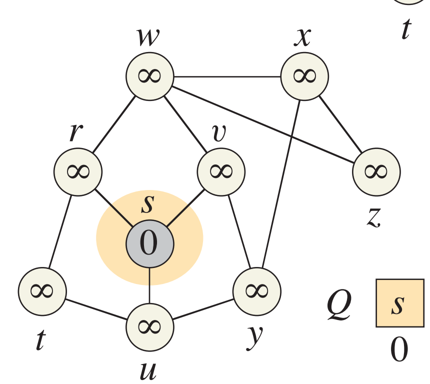
Breadth First Search

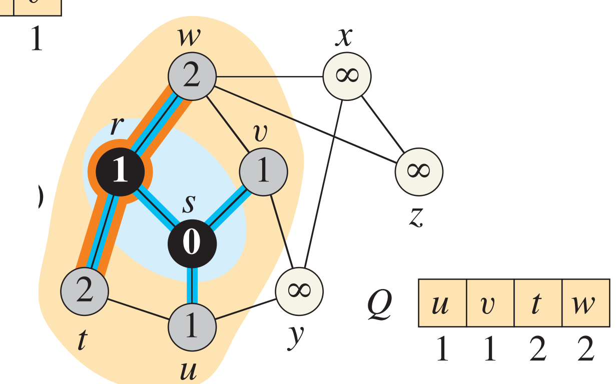
- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.

- When a node is initially discovered, it is added to the queue.

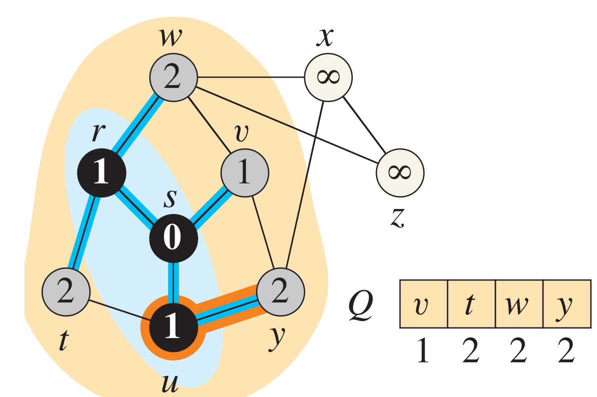


It's distance to the source node is recorded.

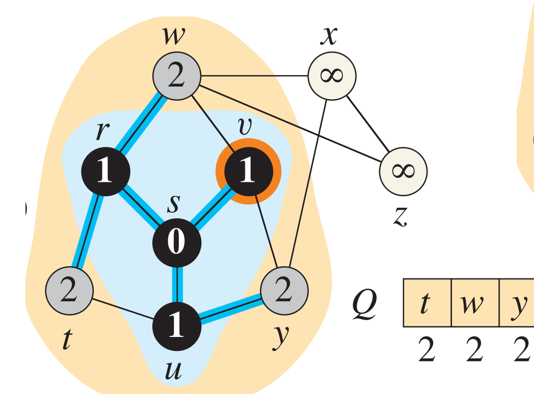




And the node is colored, grey.

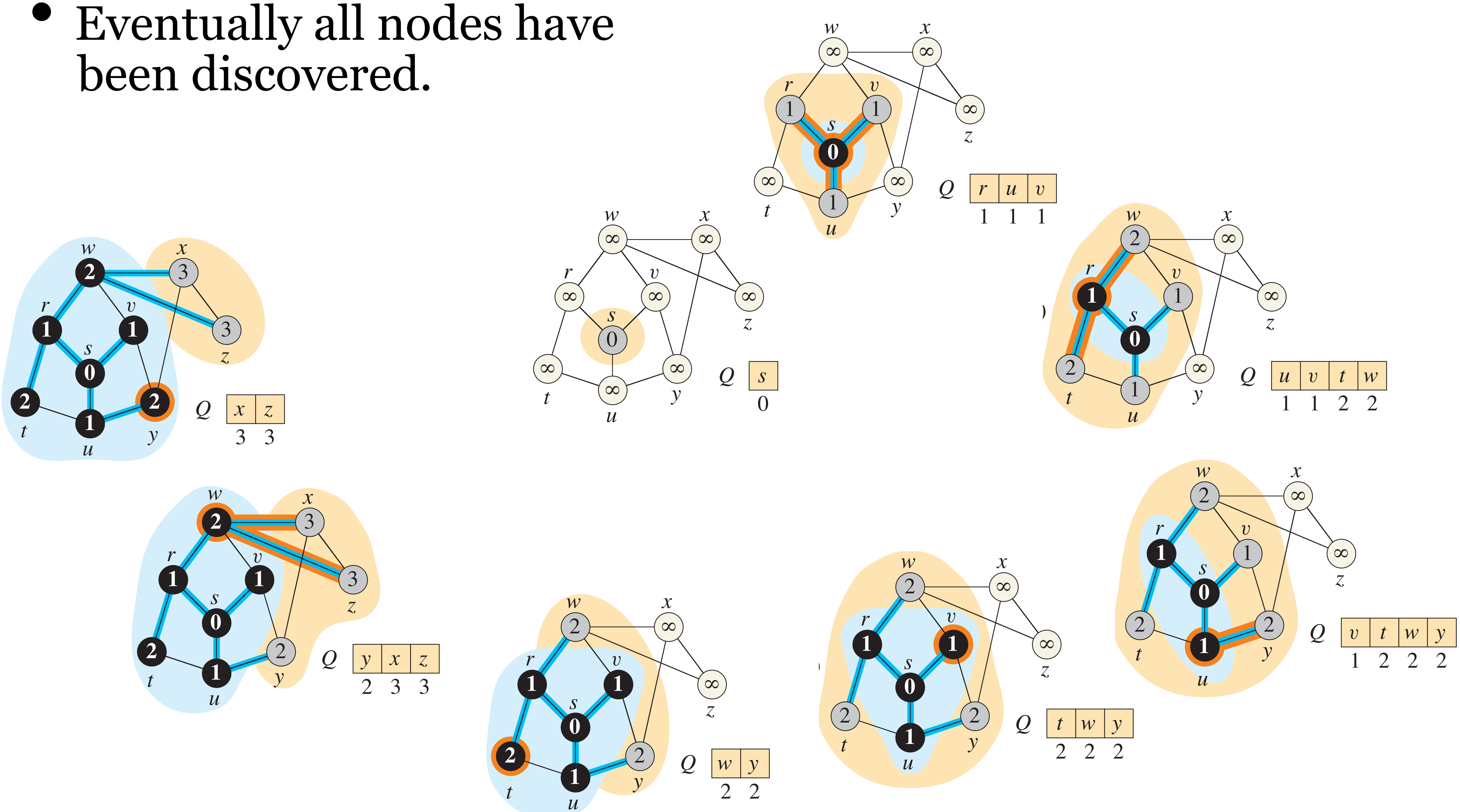


Once the node is dequeued, and it's neighbors discovered, the node is colored black.



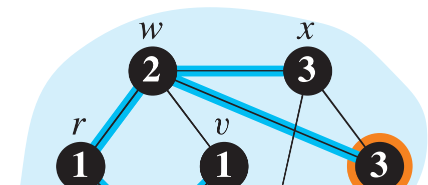
Breadth First Search

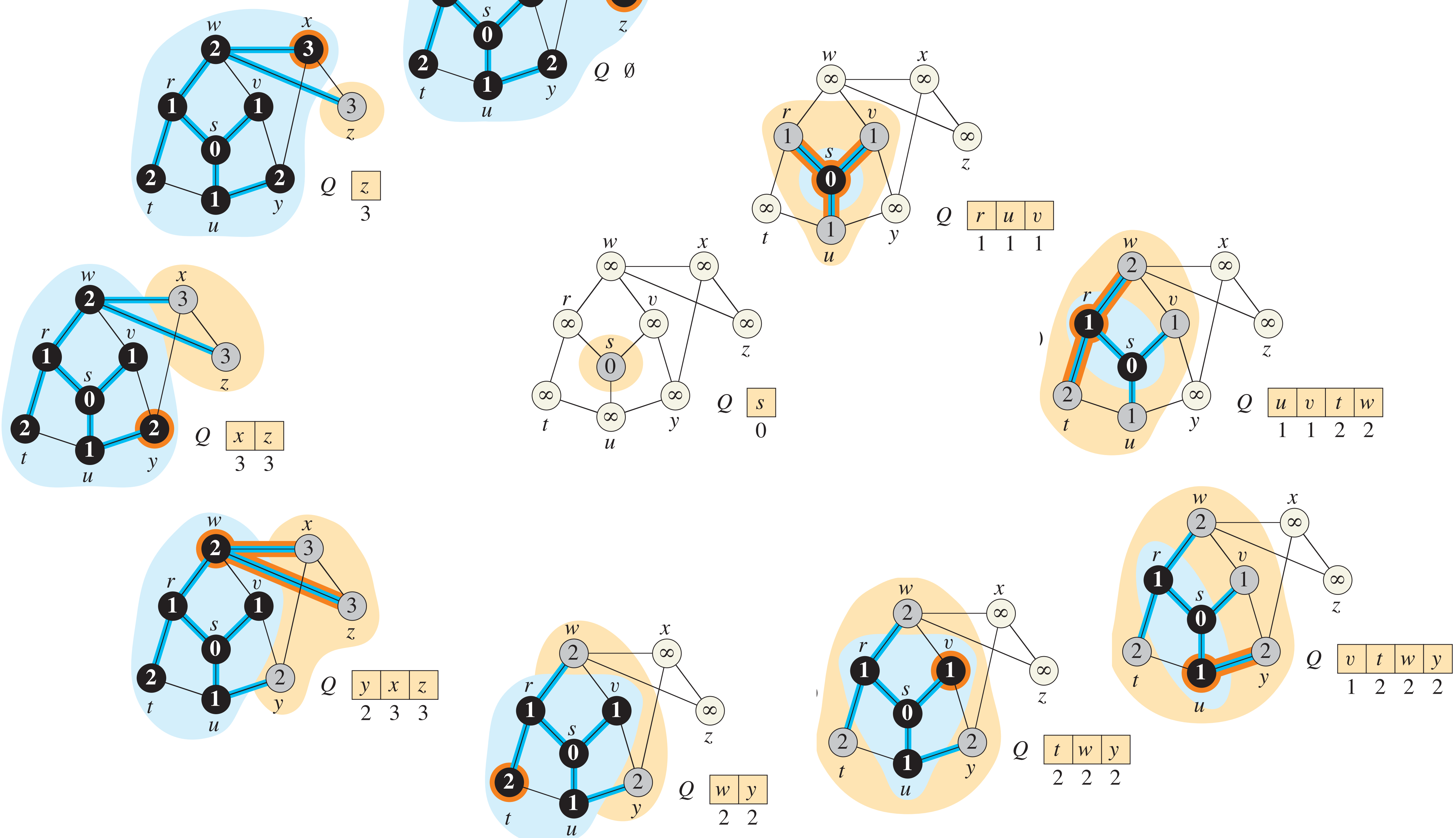
- Given a node, *s*, in a graph, we would like to know the distance to all other nodes of the graph.
- Eventually all nodes have been discovered.



Breadth First Search

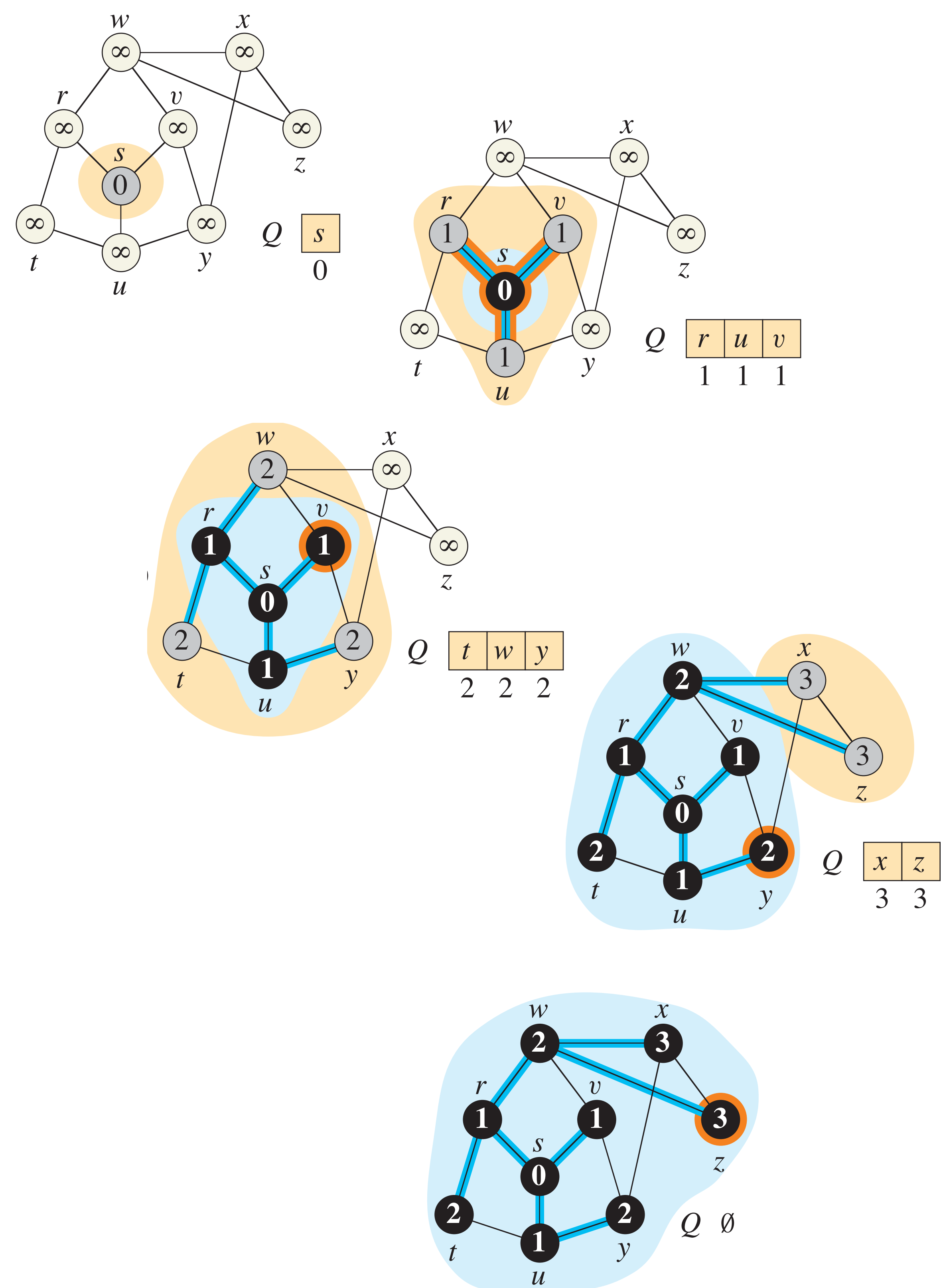
- The algorithm finishes when the queue is fully empty.





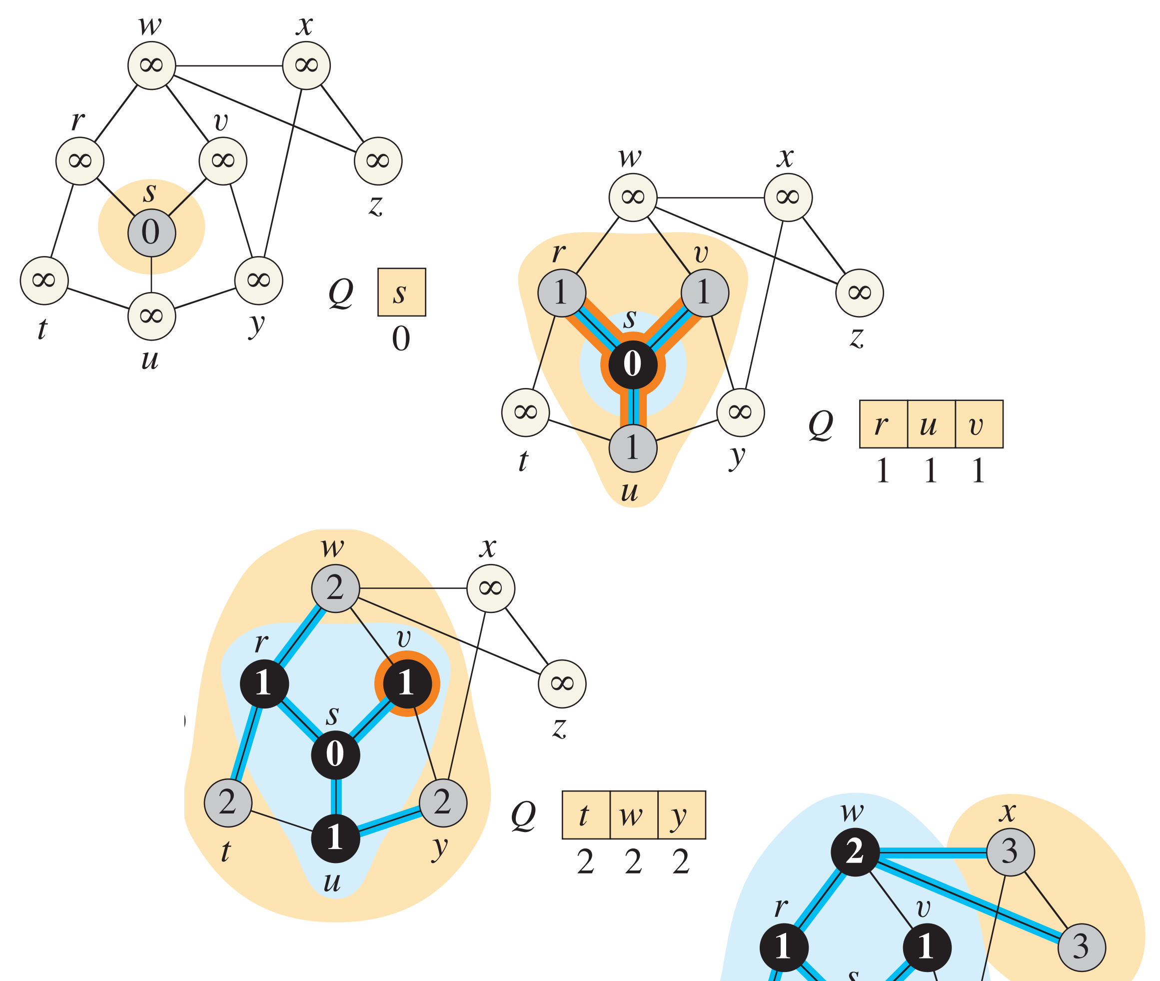
Breadth First Search

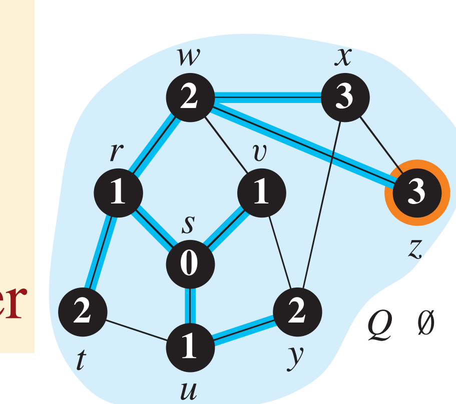
```
BFS(G, s)
    for each vertex u ∈ G.V − {s}
        u.color = WHITE
        u.d = ∞
        u.π = NIL
    s.color = GRAY
    s.d = 0
    s.π = NIL
    Q = ∅
    ENQUEUE(Q, s)
    while Q ≠ ∅
        u = DEQUEUE(Q)
        for each vertex v in G.Adj[u]
            if v.color == WHITE
            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



Breadth First Search

```
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 vertex v in G.Adj[u]     // search the neighbors of u
        if v.color == WHITE           // is v being discovered now?
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)             // v is now on the frontier
    u.color = BLACK                   // u is now behind the frontier
```





Exercise

```
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 vertex v in G.Adj[u]   // search the neighbors of u
        if v.color == WHITE         // is v being discovered now?
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)           // v is now on the frontier
    u.color = BLACK                 // u is now behind the frontier
```

- What is the running time of BFS?

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
    - A vertex is only ENQUEUED if it it is white.

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
    - A vertex is only ENQUEUED if it it is white.
    - Vertices are only colored white at initialization.

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
  - A vertex is only ENQUEUED if it it is white.
  - Vertices are only colored white at initialization.
  - Thus vertices are only ENQUEUED - and DEQUEUED- once.

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
    - Vertices are only ENQUEUED - and DEQUEUED- once.
    - Thus queueing operations take *O(V)* time, as does initialization.

Exercise

```
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 vertex v in G.Adj[u]
            if v.color == WHITE
                v.color = GRAY
                v.d = u.d + 1
                v.π = u
                ENQUEUE(Q, v)
        u.color = BLACK
```

- What is the running time of BFS?
  - Vertices are only ENQUEUED - and DEQUEUED- once.
  - Thus queueing operations take *O(V)* time, as does initialization.
  - The adjacency list of a vertex is scanned only after DEQUEUEING.

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
  - Vertices are only ENQUEUED - and DEQUEUED- once.
  - Thus queueing operations take *O(V)* time, as does initialization.
  - The adjacency list of a vertex is scanned only after DEQUEUEING.
  - Thus the adjacency lists are scanned once per vertex.

$$\sum_{V} d(v) = 2|E|$$

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- What is the running time of BFS?
  - Vertices are only ENQUEUED - and DEQUEUED- once.
  - Thus queueing operations take *O(V)* time, as does initialization.
  - The adjacency list of a vertex is scanned only after DEQUEUEING. So that all scanning takes *O(E)* time.
  - Total running time for BFS is thus *O(V+E)*.

Homework (due with Assignment 3)

- Give an example of a directed graph, $G=(V, E)$, a source vertex s, and a subset of edges which form a tree, $E_\pi \subseteq E$, with $V(G|_{E_\pi}) = V$. Such that for each vertex, v, the unique path in the graph (V, E_π) from s to v, is a shortest path in G, yet the set of edges E_π cannot be produced by running BFS on G, no matter how the vertices are ordered in the adjacency lists.

- The diameter of a tree is defined as the largest of all distances in the tree. Give an efficient algorithm to compute the diameter of a tree, and analyze the running time of your algorithm.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- To prove the correctness of BFS, for any graph G and source vertex x, we need to show

  $$v.d = d(s,v)$$

  for any vertex v.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- To prove the correctness of BFS, for any graph G and source vertex x, we need to show
$$v.d = d(s,v)$$
for any vertex v.

- We begin by showing, a loop invariant: at any point of the algorithm, it holds that:
$$v.d \geq d(s,v)$$
for any vertex v.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex *v*.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- **Initialization:**

  For any vertex $v \neq s$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- **Initialization:**

For any vertex $v \neq s$

$$v.d = \infty > d(s,v)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex *v*.

- **Initialization:**

  For any vertex $v \neq s$:

  $$v.d = \infty > d(s, v)$$

  For vertex s :

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- **Initialization:**

For any vertex $v \neq s$:

$$v.d = \infty > d(s,v)$$

For vertex s :

$$s.d = 0 = d(s,s)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- **Initialization:**

  For any vertex $v \neq s$:

  $$v.d = \infty > d(s,v)$$

  For vertex s :

  $$s.d = 0 = d(s,s)$$

  Proving

  $$v.d \geq d(s,v)$$

  for any vertex v.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- **Maintenance:**

Assume the hypothesis still holds for all vertices.

Notice some vertices have now been colored gray, and their $.d$ attribute has been modified.

Notice a **while** iteration will only modify the $.d$ attribute of white vertices.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- **Maintenance:**

Consider a gray vertex u and let v be one of it's white neighbors.

By hypothesis,

$$u.d \geq d(s,u)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- **Maintenance:**

  Consider a gray vertex u and let v be one of it's white neighbors.

  By hypothesis,

  $$u.d \geq d(s,u)$$

  By definition:

  $$\boxed{v.d = u.d + 1} \geq d(s,u) + 1 \geq d(s,v)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- **Maintenance:**

Consider a gray vertex u and let v be one of it's white neighbors.

By hypothesis,

$$u.d \geq d(s,u)$$

Applying previous corollary:

$$\begin{aligned} v.d &= u.d + 1 \\ &\geq d(s,u) + 1 \\ &\geq d(s,v) \end{aligned}$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- **Maintenance:**

  Consider a gray vertex u and let v be one of it's white neighbors.

  By hypothesis,

  $$u.d \geq d(s,u)$$

  Applying previous corollary:

  $$\begin{aligned} v.d &= u.d + 1 \\ &\geq d(s,u) + 1 \\ &\geq d(s,v) \end{aligned}$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- **Maintenance:**

  Proving after a **while** iteration, it still holds that

  $$v.d \geq d(s,v)$$

  for any vertex v.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex *v*.

- **Termination:**

  After finishing the **while** loop, it holds that

  $$v.d \geq d(s,v)$$

  for any vertex *v*.

  Proving the loop invariant.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- To prove the correctness of BFS, for any graph G and source vertex x, we need to show

$$v.d = d(s,v)$$

  for any vertex v.

- We have shown, at any point of the algorithm, it holds that:

$$v.d \geq d(s,v)$$

  for any vertex v.

- Now let us prove a more precise property regarding the $.d$ attribute.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- We claim another loop invariant: at any point of the algorithm, if the queue, Q, contains vertices $(v_1, \ldots v_r)$ ordered such that v_1 is the head, it holds that:

$$v_r.d \le v_1.d + 1$$

$$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

  $$v_i.d \leq v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Initialization:**

  Queue contains only s, so statement holds.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

  $$v_r.d \le v_1.d + 1$$

  $$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

  During a **while** iteration, one vertex will be DEQUEUED and others will be ENQUEUED.

  Let us show DEQUEUEING maintains the loop invariant.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

  $$v_i.d \leq v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

  After DEQUEUEING, v_2 becomes the head.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

$$v_r.d \le v_1.d + 1$$

$$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After DEQUEUEING, v_2 becomes the head.

By hypothesis,

$$v_2.d \quad \ge v_1.d$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\ldots v_r)$, with v_1 the head, it holds that:

$$v_r.d \le v_1.d + 1$$

$$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After DEQUEUEING, v_2 becomes the head.

By hypothesis,

$$v_2.d + 1 \ge v_1.d + 1$$
$$\ge v_r.d$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \dots v_r)$, with v_1 the head, it holds that:

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

$$v_i . d \leq v_{i+1} . d \ \text{ for } \ i = 1, \dots, r-1$$

- **Maintenance:**

After DEQUEUEING, v_2 becomes the head.

By hypothesis,

$$v_2 . d + 1 \geq v_1 . d + 1$$

$$\geq v_r . d$$

Proving the loop invariant holds.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

  $$v_i . d \leq v_{i+1} . d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

  If the queue is empty before ENQUEUING, the loop invariant automatically holds after the ENQUEUE operation, since the queue has a single vertex.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

$$v_i . d \leq v_{i+1} . d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

If the queue is empty before ENQUEUING, the loop invariant automatically holds after the ENQUEUE operation, since the queue has a single vertex.

Assume the queue were not initially empty.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\ldots v_r)$, with v_1 the head, it holds that:

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

$$v_i.d \leq v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After ENQUEUING, the tail is now vertex v_{r+1}. We know it is a neighbor of the most recently dequeued vertex, u, which up to the last iteration satisfied:

$$v_r.d \leq u.d + 1$$

and $u.d \leq v_2.d$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\ldots v_r)$, with v_1 the head, it holds that:

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

$$v_i.d \leq v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After ENQUEUING, the tail is now vertex v_{r+1}. We know it is a neighbor of the most recently dequeued vertex, u, which this iteration satisfies:

$$v_r.d \leq u.d + 1$$

and $u.d \leq \cancel{v_2.d}\ v_1.d$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\ldots v_r)$, with v_1 the head, it holds that:

$$v_r.d \le v_1.d + 1$$

$$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After ENQUEUEING, the tail is now vertex v_{r+1}. We know it is a neighbor of the most recently dequeued vertex, u, which this iteration satisfies:

$$v_r.d \le u.d + 1$$

$$u.d \le v_1.d$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\ldots v_r)$, with v_1 the head, it holds that:

$$v_r.d \le v_1.d + 1$$

$$v_i.d \le v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Maintenance:**

After ENQUEUEING, the tail is now vertex v_{r+1}. We know it is a neighbor of the most recently dequeued vertex, u, which this iteration satisfies:

$$v_r.d \le u.d + 1 = v_{r+1}.d$$

$$v_{r+1}.d = u.d + 1 \le v_1.d + 1$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1,\dots v_r)$, with v_1 the head, it holds that:

  $$v_r.d \le v_1.d + 1$$

  $$v_i.d \le v_{i+1}.d \text{ for } i = 1,\dots,r-1$$

- **Maintenance:**

  After ENQUEUING, the tail is now vertex v_{r+1}. We know it is a neighbor of the most recently dequeued vertex, u, which this iteration satisfies:

  $$v_r.d \le v_{r+1}.d$$

  $$v_{r+1}.d \le v_1.d + 1$$

  Proving the loop invariant.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

$$v_i . d \leq v_{i+1} . d \text{ for } i = 1, \ldots, r-1$$

- **Termination:**

  After finishing the **while** loop, the loop invariant holds as the queue is empty.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- **Loop invariant**: if the queue $Q=(v_1, \ldots v_r)$, with v_1 the head, it holds that:

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

$$v_i.d \leq v_{i+1}.d \text{ for } i = 1,\ldots,r-1$$

- **Corollary:**

If v_i is enqueued before v_j, then

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

when v_j is enqueued.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- To prove the correctness of BFS, for any graph G and source vertex x, we need to show

  $$v.d = d(s,v)$$

  for any vertex v.

- We have shown, at any point of the algorithm, it holds that:

  $$v.d \geq d(s,v)$$

  for any vertex v.

- And that if v_i is enqueued before v_j, then

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

  when v_j is enqueued.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- We have shown, at any point of the algorithm, it holds that:

$$v.d \geq d(s,v)$$

for any vertex v.

- And that if v_i is enqueued before v_j, then

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

when v_j is enqueued.

- Let us now prove the correctness of the algorithm, by contradiction. Assume there is a vertex v such that:

$$v\,.\,d > d(s, v)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s, v)$$

That has smallest $d(s,v)$ value.

Correctness of BFS

```
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 vertex v in G.Adj[u]
            if v.color == WHITE
                v.color = GRAY
                v.d = u.d + 1
                v.π = u
                ENQUEUE(Q, v)
        u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s, v)$$

That has smallest $d(s,v)$ value.

  - Notice $v \neq s$, or both values would be zero.
  - And there must exist some path connecting v and s, or $d(s, v) = \infty$, larger than any value.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s, v)$$

That has smallest $d(s,v)$ value.

  - Notice $v \neq s$, or both values would be zero.
  - And there must exist some path connecting v and s, or $d(s, v) = \infty$, larger than any value.
  - Thus there must exist some shortest path, of length at least one. Let u be the vertex preceding v on this path.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v\,.\,d > d(s, v)$$

That has smallest *d(s,v)* value.

- Let u be the vertex preceding v on a shortest path from s to v.
- By construction:

$$d(s, v) = d(s, u) + 1$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s, v)$$

That has smallest $d(s,v)$ value.

- Let u be the vertex preceding v on a shortest path from s to v.
- By construction:

$$d(s, v) = d(s, u) + 1$$

- By hypothesis, $d(s, u) = u.d$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s, v)$$

That has smallest $d(s,v)$ value.

- Let u be the vertex preceding v on a shortest path from s to v.
- By construction:

$$d(s, v) = d(s, u) + 1$$

- By hypothesis, $d(s, u) = u.d$
- Yielding:

$$v.d > d(s, v) = u.d + 1$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s,v)$$

That has smallest $d(s,v)$ value.

- Then v has a neighbor u, such that:

$$v.d > u.d + 1 \qquad (*)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Specifically, consider the vertex v satisfying:

$$v.d > d(s,v)$$

That has smallest $d(s,v)$ value.

- Then v has a neighbor u, such that:

$$v.d > u.d + 1 \qquad (*)$$

- Consider the time when u has just been DEQUEUED. We will show wether v is white, gray or black, $(*)$ implies a contradiction.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is white.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is white.
- Then the algorithm sets:

$$v.d = u.d + 1$$

Contradicting our hypothesis.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is black.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is black.
- This means v was ENQUEUED before u.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is black.
- This means v was ENQUEUED before u.
- Earlier we proved:

  If v is ENQUEUED before u, then
  $$v.d \le u.d$$
  when u is ENQUEUED.

- Contradicting our hypothesis.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is gray.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is gray.
- Then there exists some other vertex w, that was ENQUEUED before u, such that:

$$v.d = w.d + 1$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is gray.
- Then there exists some other vertex w, that was ENQUEUED before u, such that:

$$v.d = w.d + 1$$

- As we just recalled though:

If w is ENQUEUED before u, then

$$w.d \leq u.d$$

when u is ENQUEUED.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- Assume v has a neighbor u, such that:

$$v.d > u.d + 1$$

- Let it be the time when u has just been DEQUEUED, assume v is gray.
- Then there exists some other vertex w, that was ENQUEUED before u, such that:

$$v.d = w.d + 1$$

- As we just recalled though:

  If w is ENQUEUED before u, then
  $$w.d \leq u.d$$
  when u is ENQUEUED.

- Implying: $v.d \leq u.d + 1$ **!!!**

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- We've showed that if we assume:
  - There exists a vertex *v* satisfying:

    $$v.d > d(s, v)$$

  - And choose the specific vertex with smallest *d(s,v)* value.
  - Then this vertex *v* must have a neighbor *u*, such that:

    $$v.d > u.d + 1$$

  - Which yields a contradiction.

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- We've showed that if we assume:
  - There exists a vertex v satisfying:

$$v.d > d(s, v)$$

  - And choose the specific vertex with smallest $d(s,v)$ value.
  - Then this vertex v must have a neighbor u, such that:

$$v.d > u.d + 1$$

  - Which yields a contradiction.
- Thus proving for all vertices,

$$v.d \leq d(s, v)$$

Correctness of BFS

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- In conclusion we have proven both:

$$v.d \leq d(s, v)$$

$$v.d \geq d(s, v)$$

  for any vertex v.

- Implying: $v.d = d(s, v)$ for all vertices v in the graph, proving the correctness of the algorithm.

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- After running BFS, how can you form a shortest path from the source s to any vertex v?

Exercise

```
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 vertex v in G.Adj[u]
        if v.color == WHITE
            v.color = GRAY
            v.d = u.d + 1
            v.π = u
            ENQUEUE(Q, v)
    u.color = BLACK
```

- After running BFS, how can you form a shortest path from the source s to any vertex v?

- Construct the path recursively, by taking the shortest path from s to $v.\pi$, and adding the edge $(v.\pi, v)$.