```
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 = Ø
ENQUEUE(Q, s)
while Q ≠ Ø
    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
```

INITIALIZE-SINGLE-SOURCE(G,s)

- Graph represented as adjacency list
- π – predecessor vertex in BFS tree
- d – distance from source
- Q – a FIFO queue

RELAX(u,v)

```
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
```

- Graph represented as adjacency list
- π – predecessor vertex in BFS tree
- d – distance from source
- Q – a FIFO queue

RUNNING-TIME ANALYSIS

- Initialization: O(V)
- Queue: each vertex enqueued/ dequeued once – O(V)
- Each adj. list is scanned once (when vertex is dequeued) – O(E)

For a total running time of **O(V+E)**

Single-source Shortest Paths

- $G = (V, E)$, with $w : E \to \mathbb{R}$.
- $p = \langle v_0, v_1, \ldots, v_k \rangle$ is a ***path*** if $(v_{i-1}, v_i) \in E$ for each $i, 1 \le i \le k$.
- The ***weight*** of p:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- The ***shortest-path weight*** $\delta(u, v)$:

$$\delta(u,v) = \begin{cases} \min\{w(p) \ : \ u \overset{p}{\leadsto} v\} & \text{if there is a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- The ***shortest path*** from u to v is a path p, $u \overset{p}{\leadsto} v$, with weight $w(p) = \delta(u, v)$

Single-source Shortest Paths: Dijkstra’s Algorithm

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Q – a priority queue

S – vertices to which the shortest path has been discovered

Single-source Shortest Paths: Dijkstra's Algorithm

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

Proved correct using loop invariant

At the start of the while loop

$v.d = \delta(s, v)$ for each vertex $v \in S$.

Single-source Shortest Paths: Dijkstra's Algorithm

```
INITIALIZE-SINGLE-SOURCE(G, s)
for each vertex v ∈ G.V
    v.d = ∞
    v.π = NIL
s.d = 0
```

```
RELAX(u, v, w)
if v.d > u.d + w(u, v)
    v.d = u.d + w(u, v)
    v.π = u
```

```
DIJKSTRA(G, w, s)
INITIALIZE-SINGLE-SOURCE(G, s)
S = ∅
Q = G.V
while Q ≠ ∅
    u = EXTRACT-MIN(Q)
    S = S ∪ {u}
    for each vertex v ∈ G.Adj[u]
        RELAX(u, v, w)
```

RUNNING-TIME ANALYSIS

- Each vertex enqueued/ dequeued once
- Each adj. list is scanned once (when vertex is dequeued); hence, each edge looked at once