Graph Algorithms

CS3104

Dr. Samit Biswas, *Assistant Professor,*
Department of Computer Sc. and Technology,
Indian Institute of Engineering Science and Technology, Shibpur

Email: samit@cs.iiests.ac.in

Graph Traversal

- **Goal:** visit each vertex that is reachable from some starting vertex
- **And:** even if there are many paths to a node, visit only once
- Two algorithms:
  - DFS
  - BFS

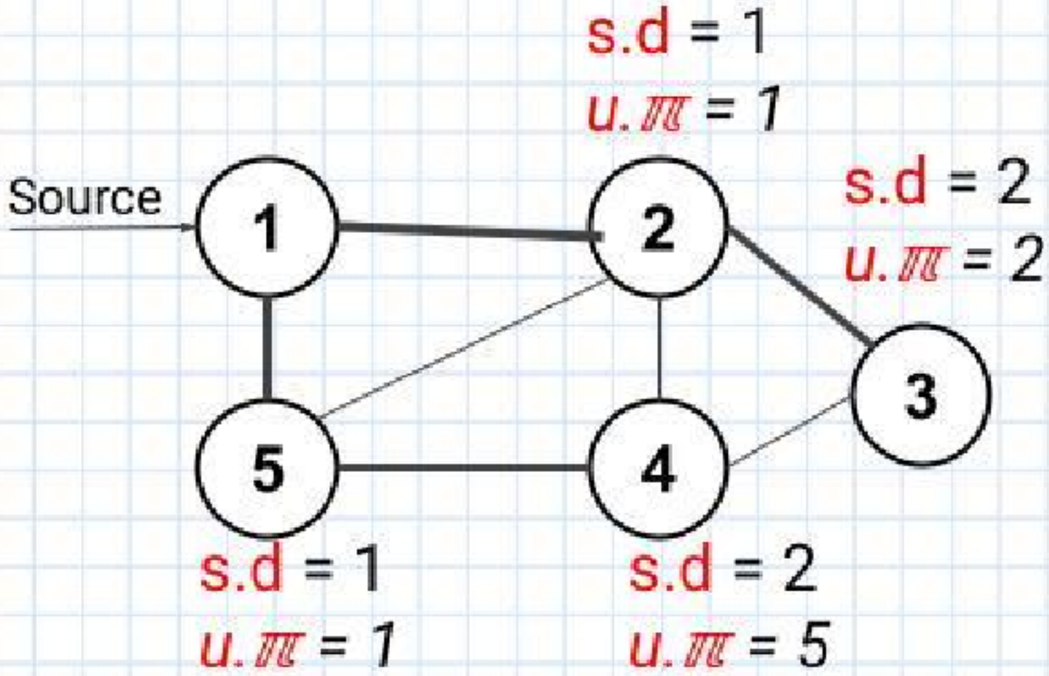
- **Applications**
  - Useful when we want to find minimal steps to reach a node

Breadth-First Search (BFS)

- **Input:**
  - A graph G = (V, E) (directed or undirected)
  - A **source** vertex **s** from V
- **Goal:**
  - Explore the edges of **G** to "discover" every vertex reachable from **s**, **taking the ones closest to s first**
- **Output:**
  - A "breadth-first tree" rooted at **s** that contains all reachable vertices.

BFS Implementation Details

- $G = (V, E)$ represented using adjacency lists
- $u.color$ = color of vertex u in V
- $u.\pi$ = predecessor of u
  - If u = s (root) or node u has not yet been discovered then $u.\pi$ = NIL
- u.d = distance (hop count) from source s to vertex u
- Use a FIFO queue Q to maintain set of gray vertices



BFS Algorithm

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

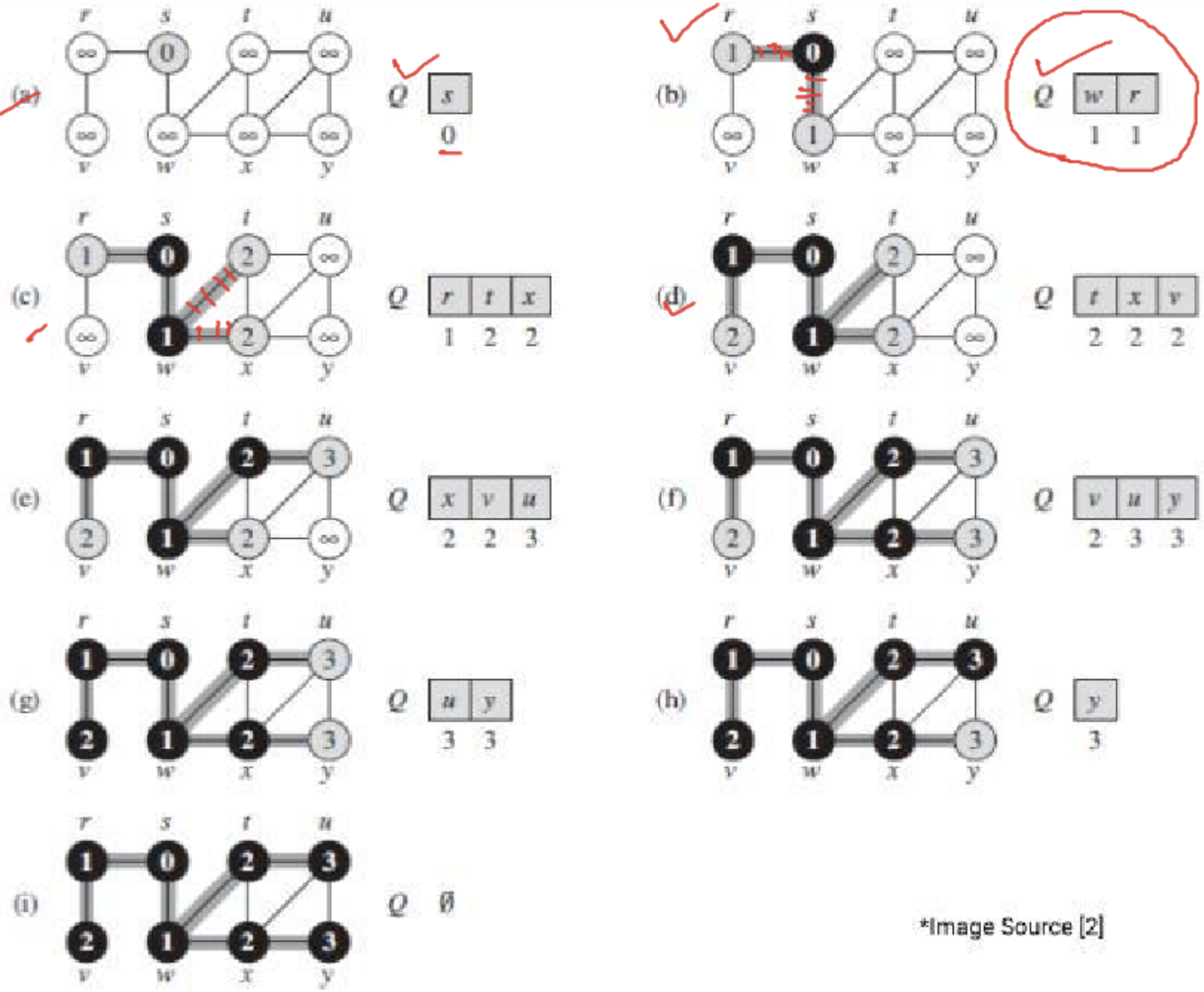


*Image Source [2]

Analysis of BFS

```
BFS(G, s)
1  for each vertex u ∈ G.V − {s}
2      u.color = WHITE
3      u.d = ∞
4      u.π = NIL
5  s.color = GRAY
6  s.d = 0
7  s.π = NIL
8  Q = ∅
9  ENQUEUE(Q, s)
10 while Q ≠ ∅
11     u = DEQUEUE(Q)
12     for each v ∈ G.Adj[u]
13         if v.color == WHITE
14             v.color = GRAY
15             v.d = u.d + 1
16             v.π = u
17             ENQUEUE(Q, v)
18     u.color = BLACK
```

O(|V|)

Θ(1)

Θ(1)

Θ(1)

Total running time for BFS = O(|V| + |E|)

O(V + E)

- Scan G.Adj[u] for all vertices u in the graph
- Each vertex u is processed only once, when the vertex is dequeued
- Sum of lengths of all adjacency lists = Θ(|E|)
- Scanning operations: O(|E|)

Shortest path property

- BFS finds the shortest-path distance from the source vertex $s \in V$ to each node in the graph
- Shortest-path distance = $\delta(s, u)$
  - Minimum number of edges in any path from s to u
  - if there is no path from s to u, then $\delta(s, u) = \infty$

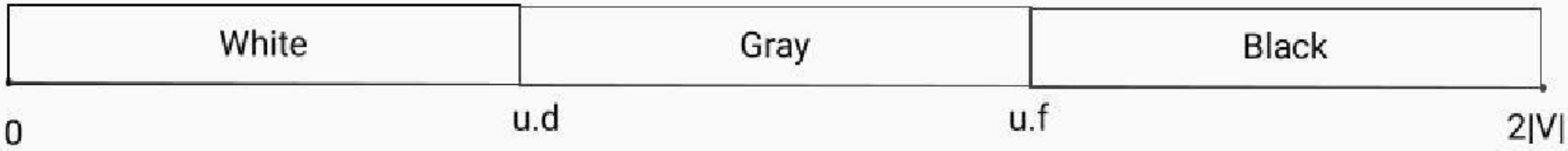
Shortest path property

- **Input:**
  - A graph G = (V, E) (directed or undirected)
  - No source vertex given!
- **Goal:**
  - Explore edges of **G** to “discover” every vertex in V starting at **most current visited node**
- **Output:**
  - v.d = discovery time (time when v is first reached and grayed)
  - v.f = finishing time (done with examining v’s adjacency list and blackens v)
  - Depth-first forest

DFS Implementation Details

- **Global variable:** time-step
  - Incremented when nodes are discovered/finished
- u.color – color of node u
  - White not discovered, gray discovered and being processing and black when finished processing
- $u.\pi$ = predecessor of u (from which node we discover u)
- u.d = discovery time (time when v is first reached and grayed)
- u.f = finishing time (done with examining v's adjacency list and blackens v)

$$1 \le u.d \le u.f \le 2|V|$$



Shortest path property

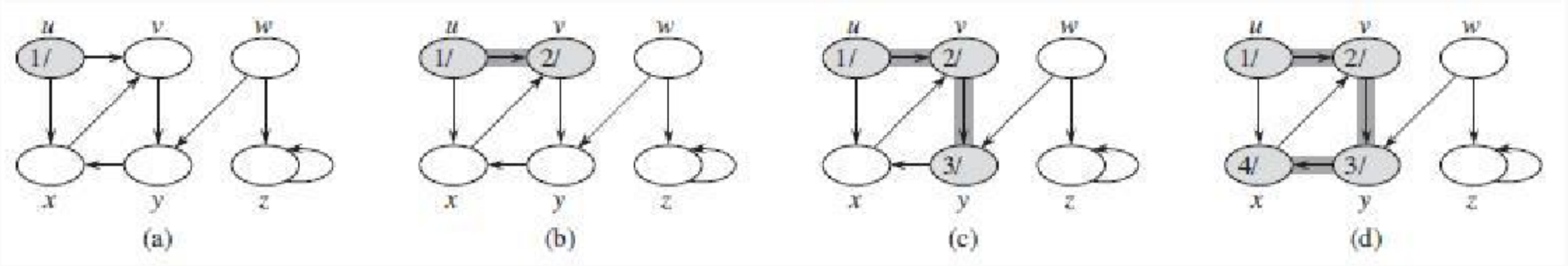
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

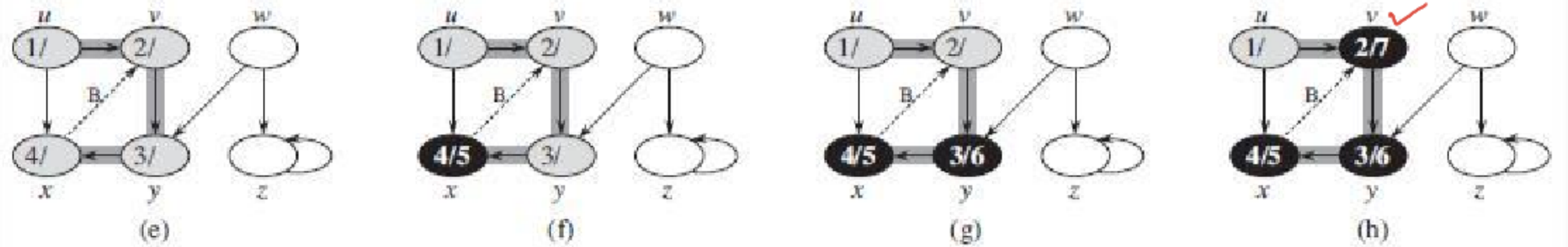
- Every time DFS-VISIT(G, u) is called, u becomes the root of a new tree in the depth-first forest

Shortest path property

```
DFS-VISIT(G, u)
 1  time = time + 1          // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]    // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK          // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

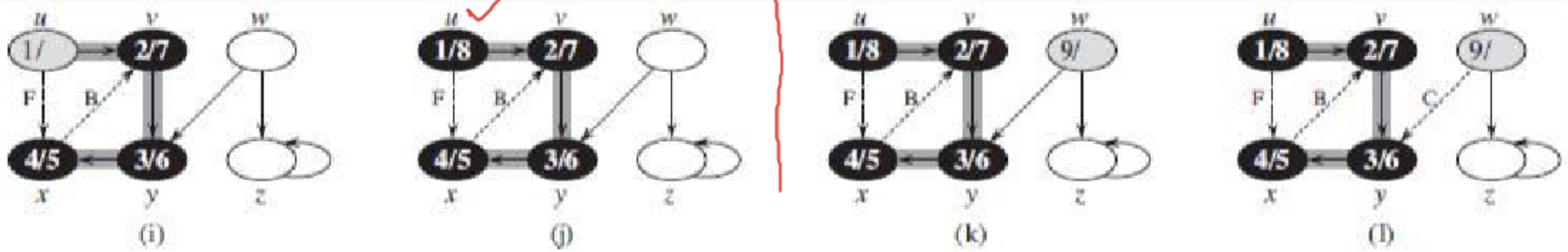
DFS - Example

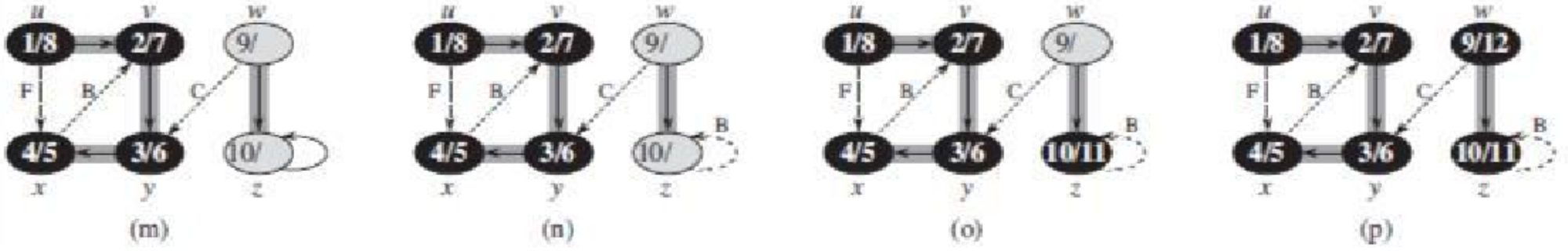




DFS - Example

Any undiscovered vertex ?





Analysis of DFS(G)

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

→ $\Theta(|v|)$

- $\Theta(|v|)$ - without counting the time for DFS-VISIT

Analysis of DFS-VISIT(G,u)

```
DFS-VISIT(G, u)
 1  time = time + 1                  // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]            // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK                  // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

$|Adj[u]|$

$\sum_{u \in V} |Adj[u]| = \Theta(E)$

Total running time of DFS

$\Theta(V + E)$