BFS & DFS

Kuan-Yu Chen (陳冠宇)

2019/05/21 @ TR-310-1, NTUST

Levenshtein Distance

- Word error rate (WER) is a common metric of the performance of a speech recognition or machine translation system
  - If the answer is “今天 天氣 很好”
    - “今天 天晴 很好”

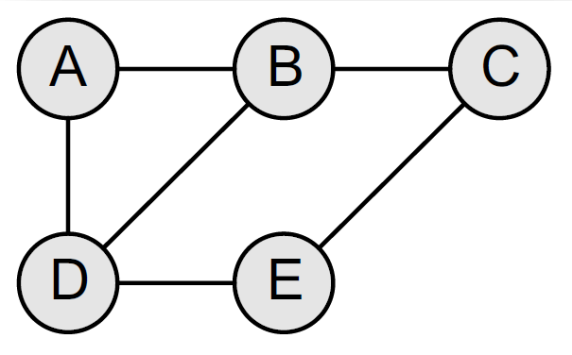
      Substitution Error
    - “今天 天氣 是 很好”

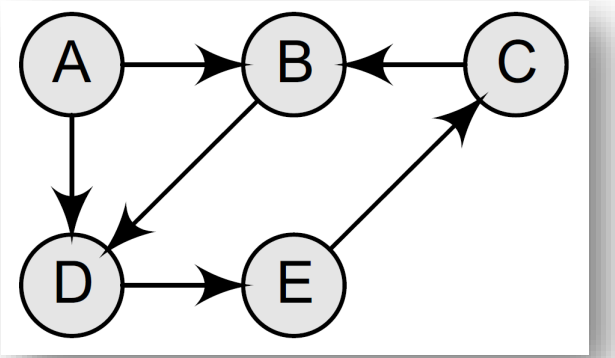
      Insertion Error
    - “今天 很好”

      Deletion Error
  - The formulation of word error rate is $WER = \frac{S+D+I}{|Ref|}$

Undirected & Directed Graphs

- A graph G is defined as an ordered set (V, E), where $V(G)$ represents the set of vertices and $E(G)$ represents the edges
  - For a given undirected graph with $V(G) = \{A, B, C, D, E\}$ and $E(G) = \{(A, B), (B, C), (A, D), (B, D), (D, E), (C, E)\}$
    - Five vertices or nodes and six edges in the graph

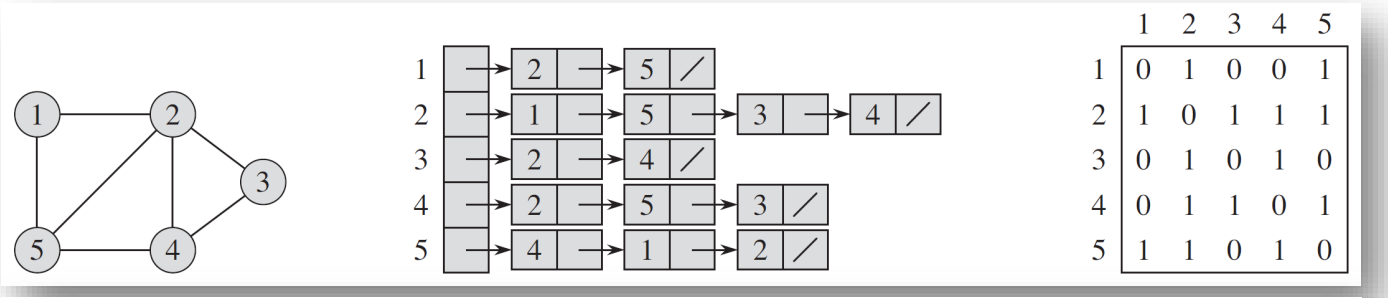


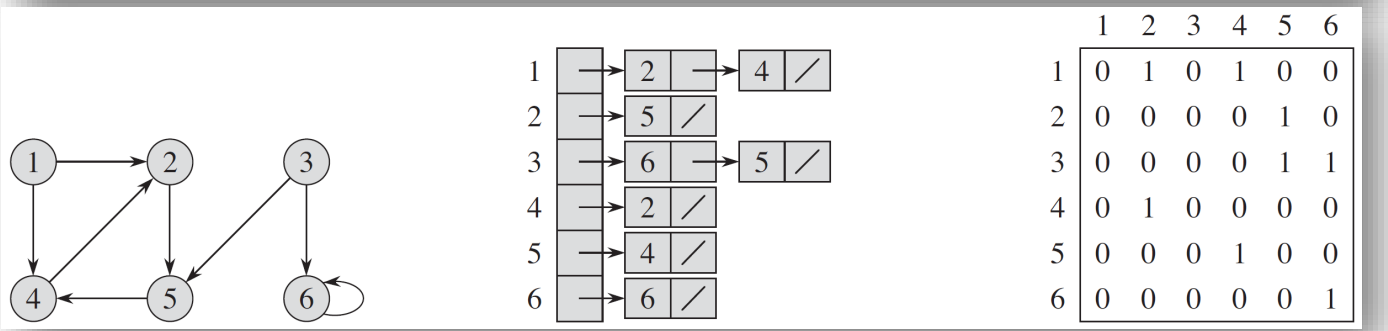


  - For a given directed graph, the edge (A, B) is said to initiate from node A (also known as initial node) and terminate at node B (terminal node)

Graph Representations

- There are two standard ways to represent a graph $G = (V, E)$
  - Adjacency list
    - **Sparse** graph
    - $|E|$ is much less than $|V|^2$
  - Adjacency matrix
    - **Dense** graph
    - $|E|$ is close to $|V|^2$



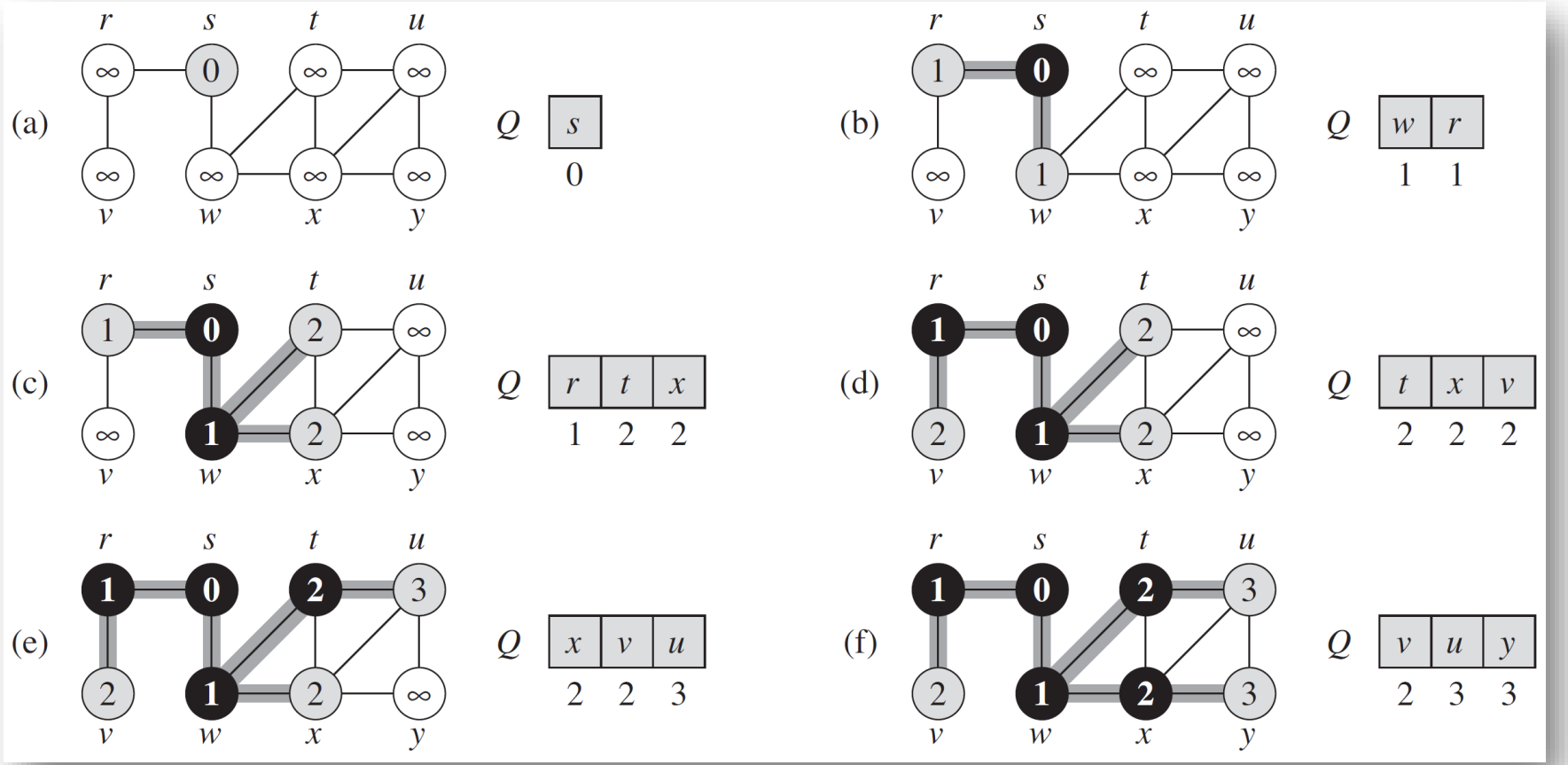


Breadth-first Search.

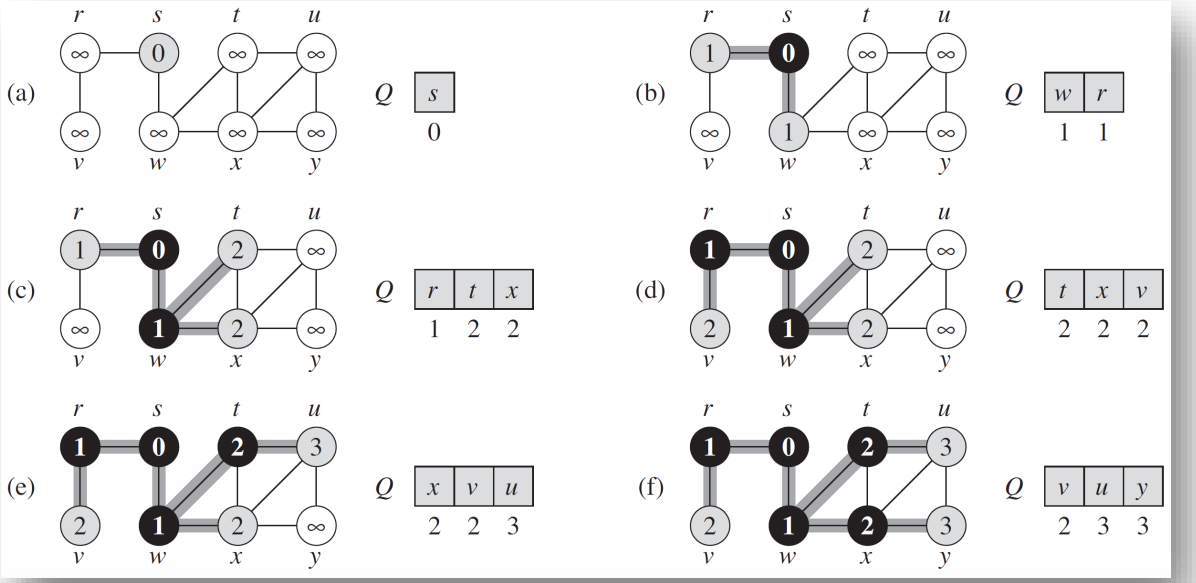
- ***Breadth-first search*** is one of the simplest algorithms for searching a graph
  - Given a graph $G = (V, E)$ and a distinguished ***source*** vertex s
    - Breadth-first search systematically explores the edges of G to "discover" every vertex that is reachable from s

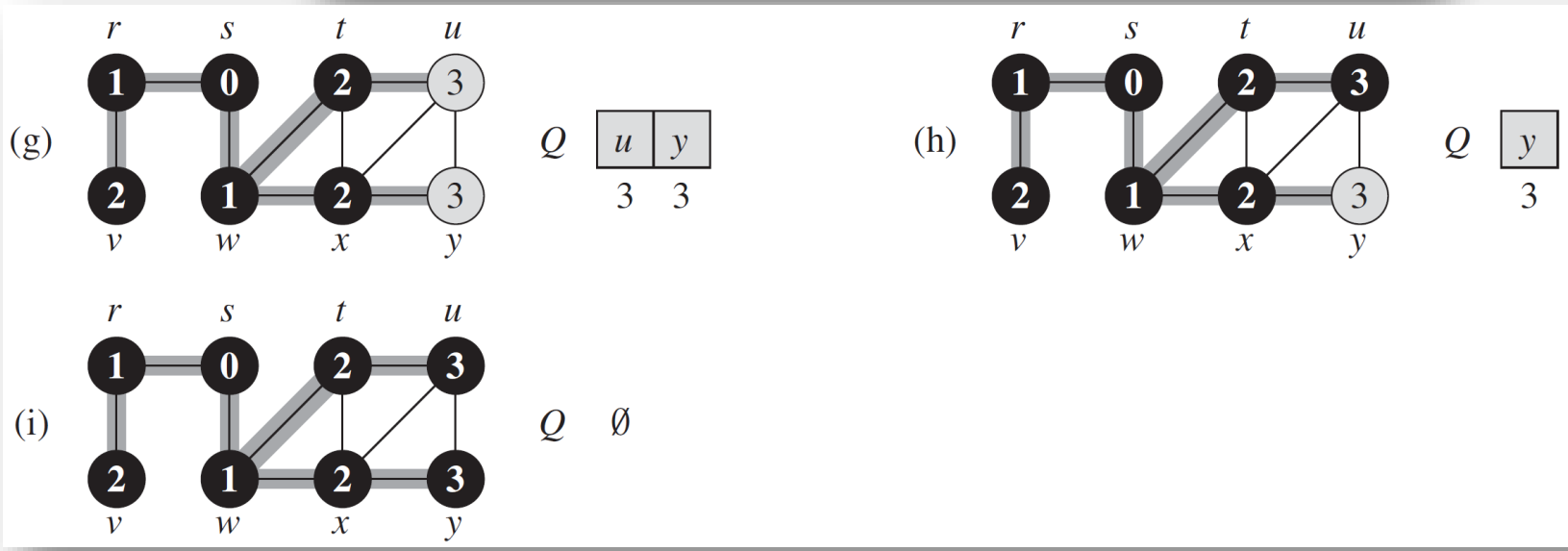
```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2       u.color = WHITE
 3       u.d = ∞
 4       u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11       u = DEQUEUE(Q)
12       for each v ∈ G.Adj[u]
13            if v.color == WHITE
14           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

Breadth-first Search..



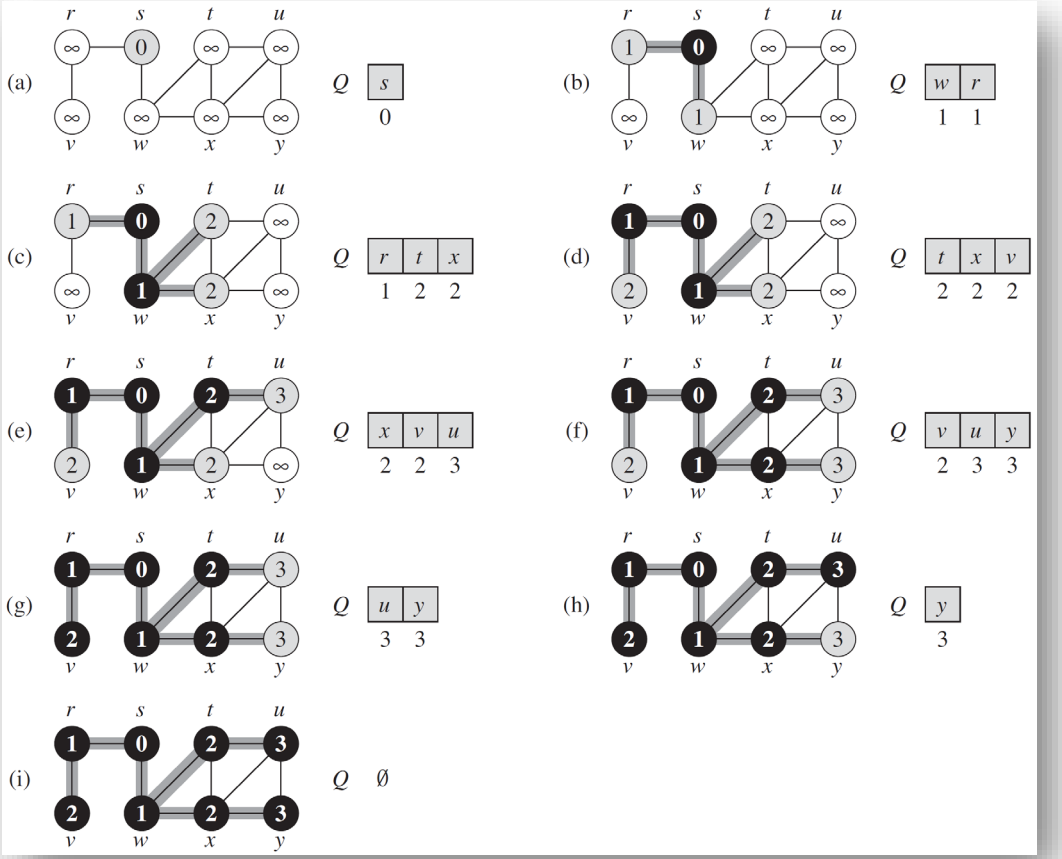
Breadth-first Search...





Breadth-first Search….

```
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
```

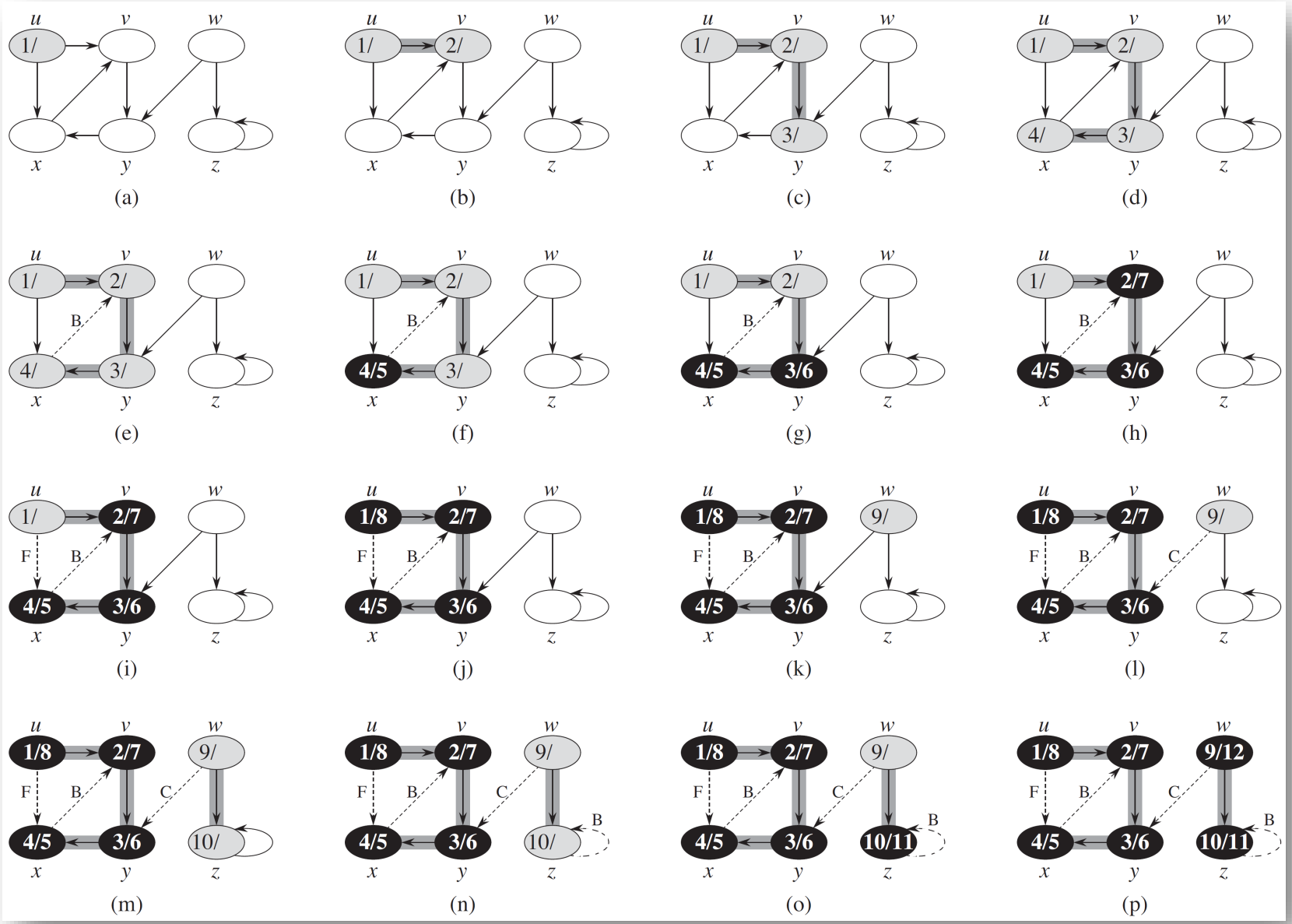


Depth-first Search.

- The strategy followed by depth-first search is, as its name implies, to search "deeper" in the graph whenever possible

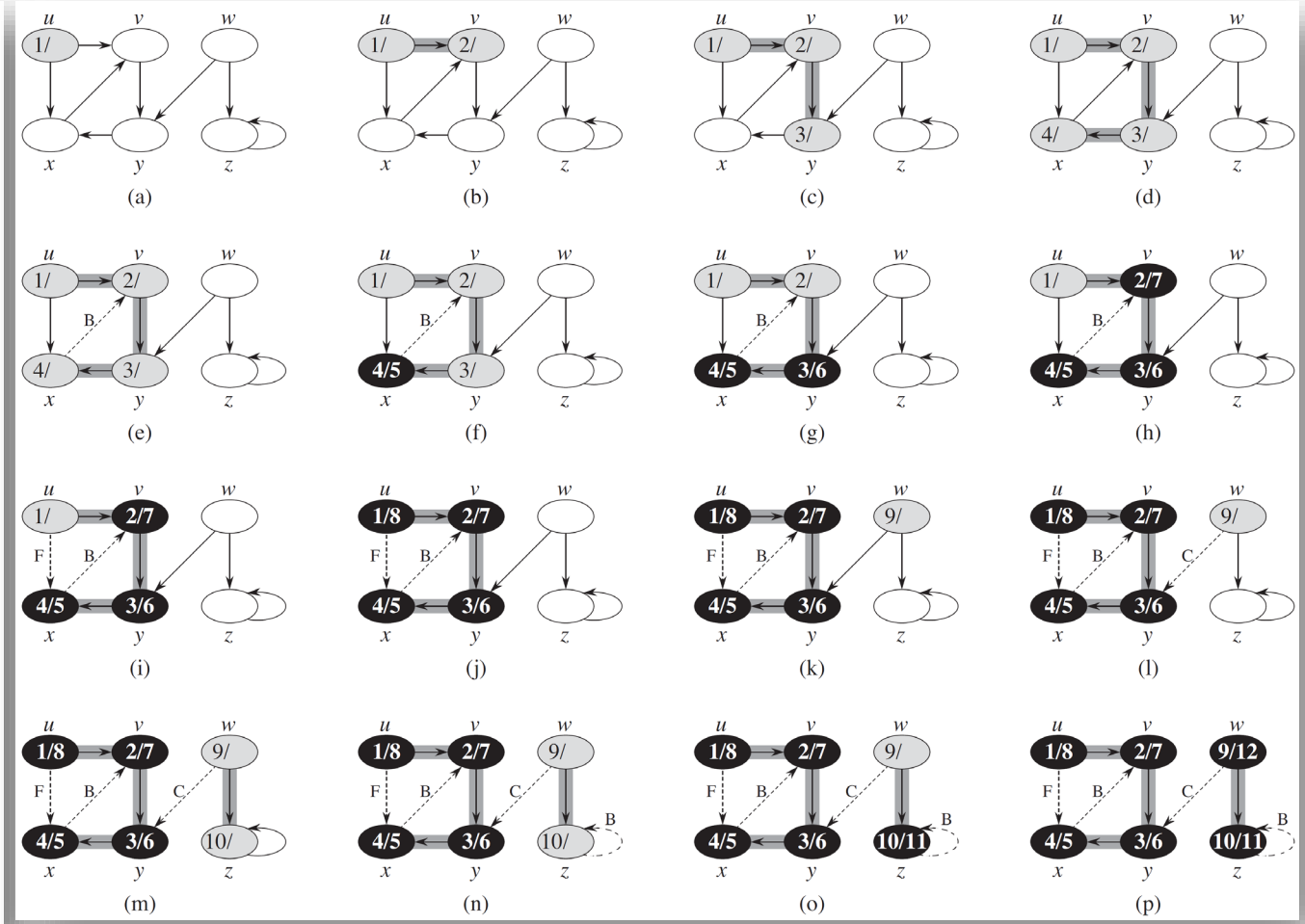
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

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
```

Depth-first Search..



Depth-first Search…

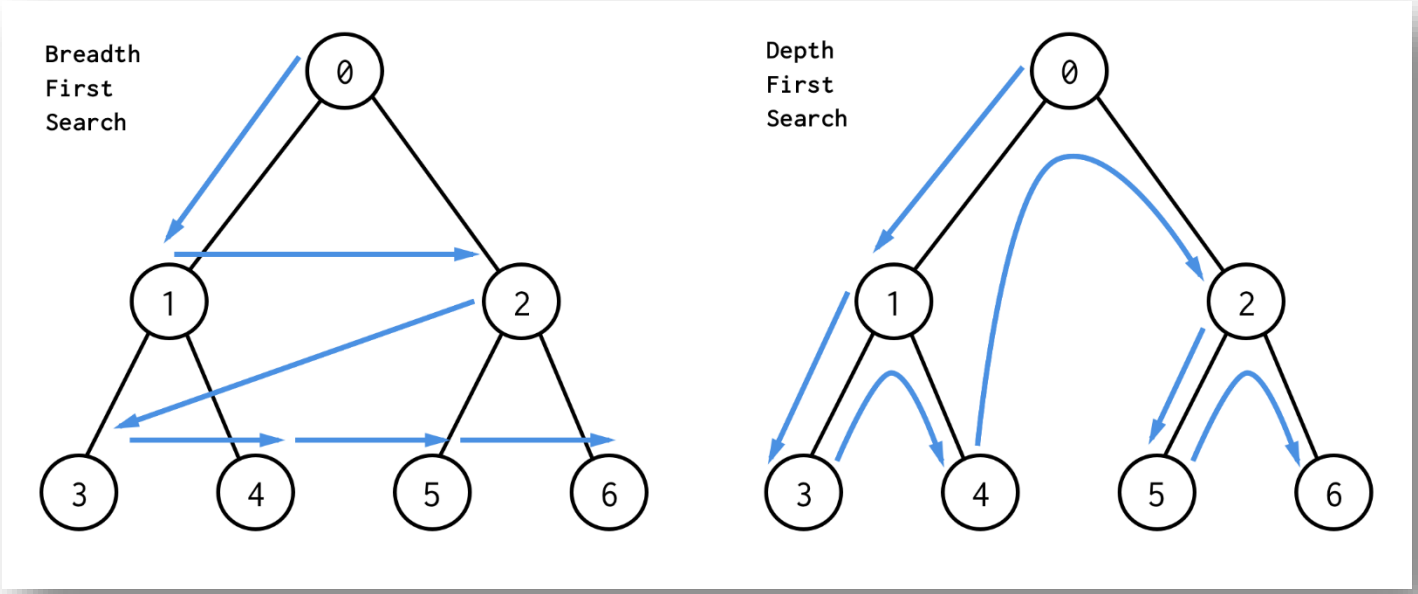
```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
 1  time = time + 1
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK
 9  time = time + 1
10  u.f = time
```



BFS & DFS

- Breadth-first search
  - BFS uses a **queue** as an auxiliary data structure to store nodes for further processing

- Depth-first search
  - DFS uses a **stack** to store nodes for further processing



https://www.quora.com/What-are-the-differences-between-DFS-and-BFS

Questions?

kychen@mail.ntust.edu.tw