# Graph Traversal Algorithms

Kuan-Yu Chen (陳冠宇)

2020/12/21 @ TR-313, NTUST

# Review

- Shortest path algorithms
  - Dijkstra's algorithm
    - Can only perform on a positive-weighted directed graph
  - Bellman-Ford algorithm
    - Can be applied on a weighted directed graph

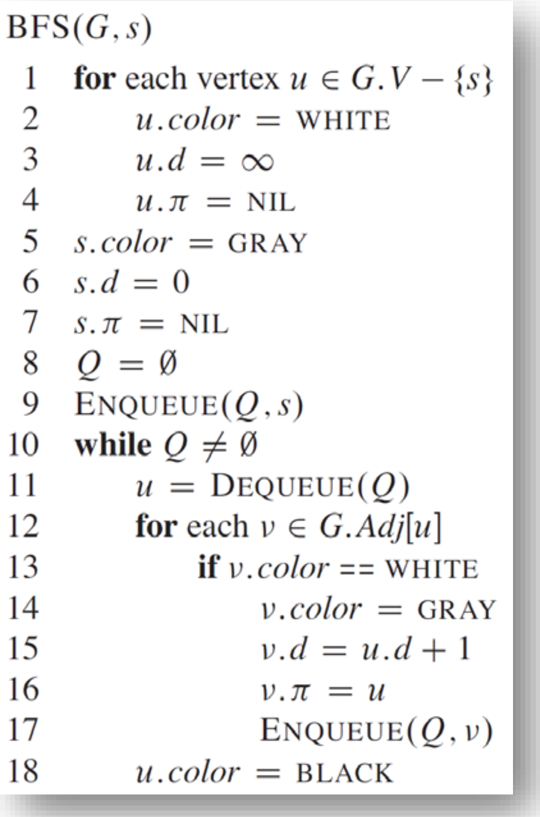
# Graph Traversals

- Several traversal algorithms have been introduced to traverse the tree structure
  - In-order
  - Pre-order
  - Post-order
  - Level-order

- An analogous situation occurs in the case of graphs
  - Given a graph $G = (V, E)$, we wish to visit all vertices in $G$
  - Breadth-first search and depth-first search are two representatives

# Breadth-first Search.

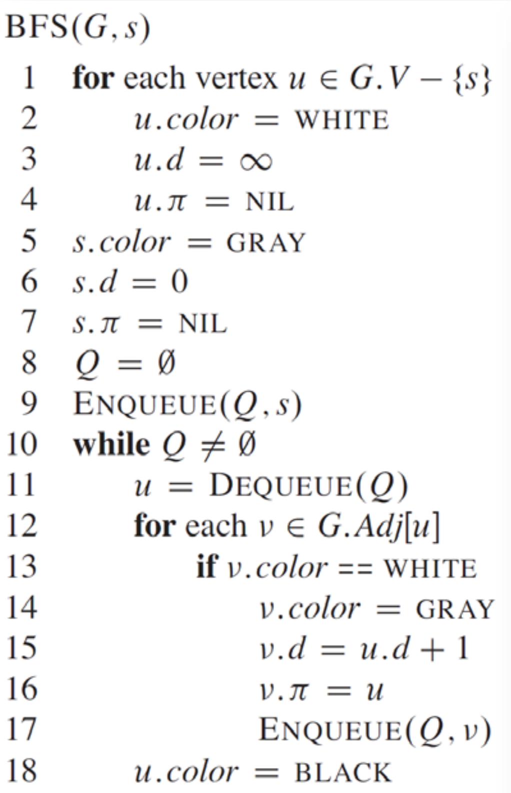
- ***Breadth-first search*** is one of the simplest algorithms for traversing a graph
  - Given a graph $G = (V, E)$ and a distinguished ***source*** vertex $s$
    - Breadth-first search systematically explores the edges of $G$ to "discover" every vertex that is reachable from $s$



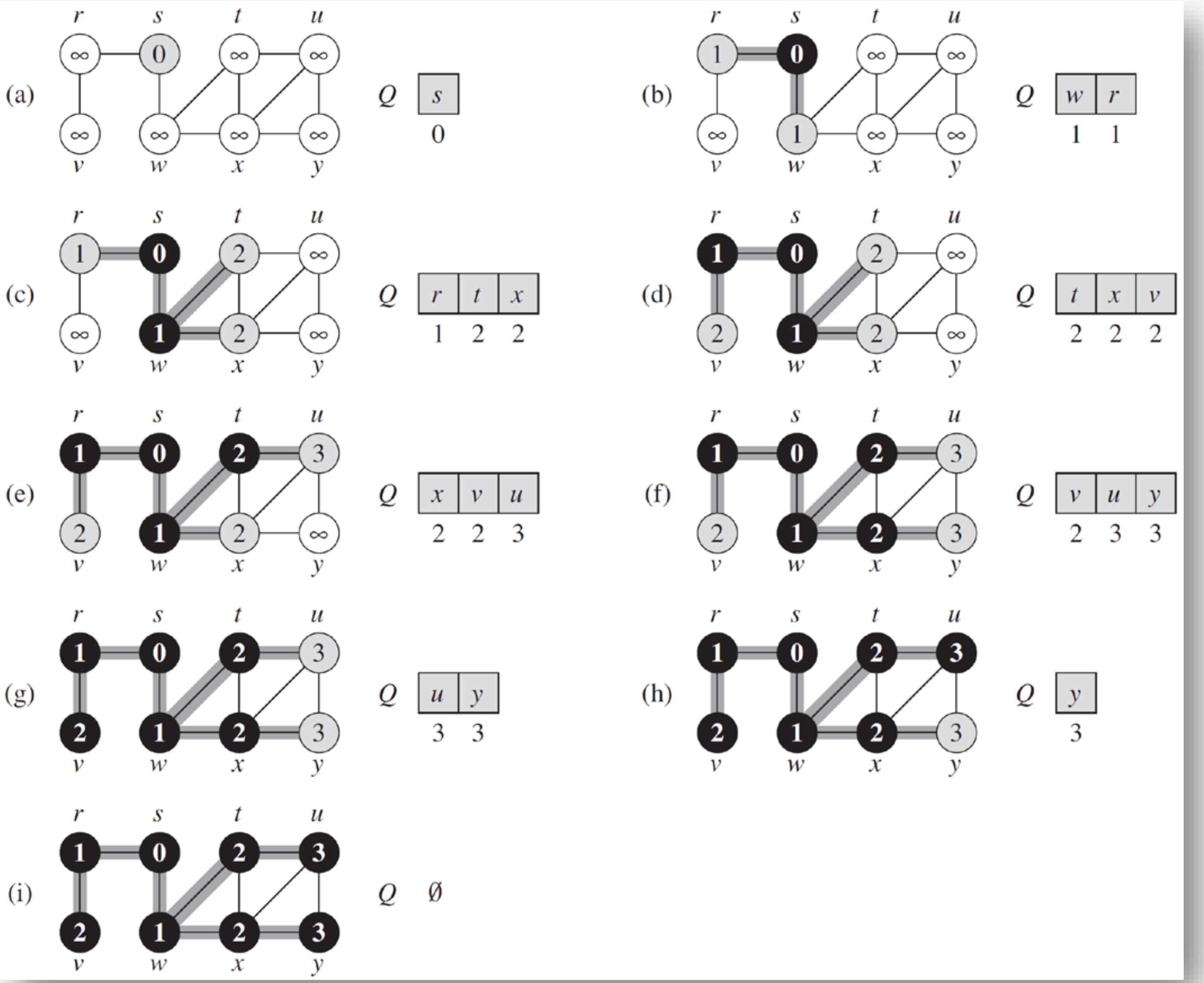
```
BFS($G, s$)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$Q = \emptyset$
ENQUEUE($Q, s$)
while $Q \neq \emptyset$
    $u$ = DEQUEUE($Q$)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
        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

# Breadth-first Search..

- The traversal sequence is “swrtxvuy”

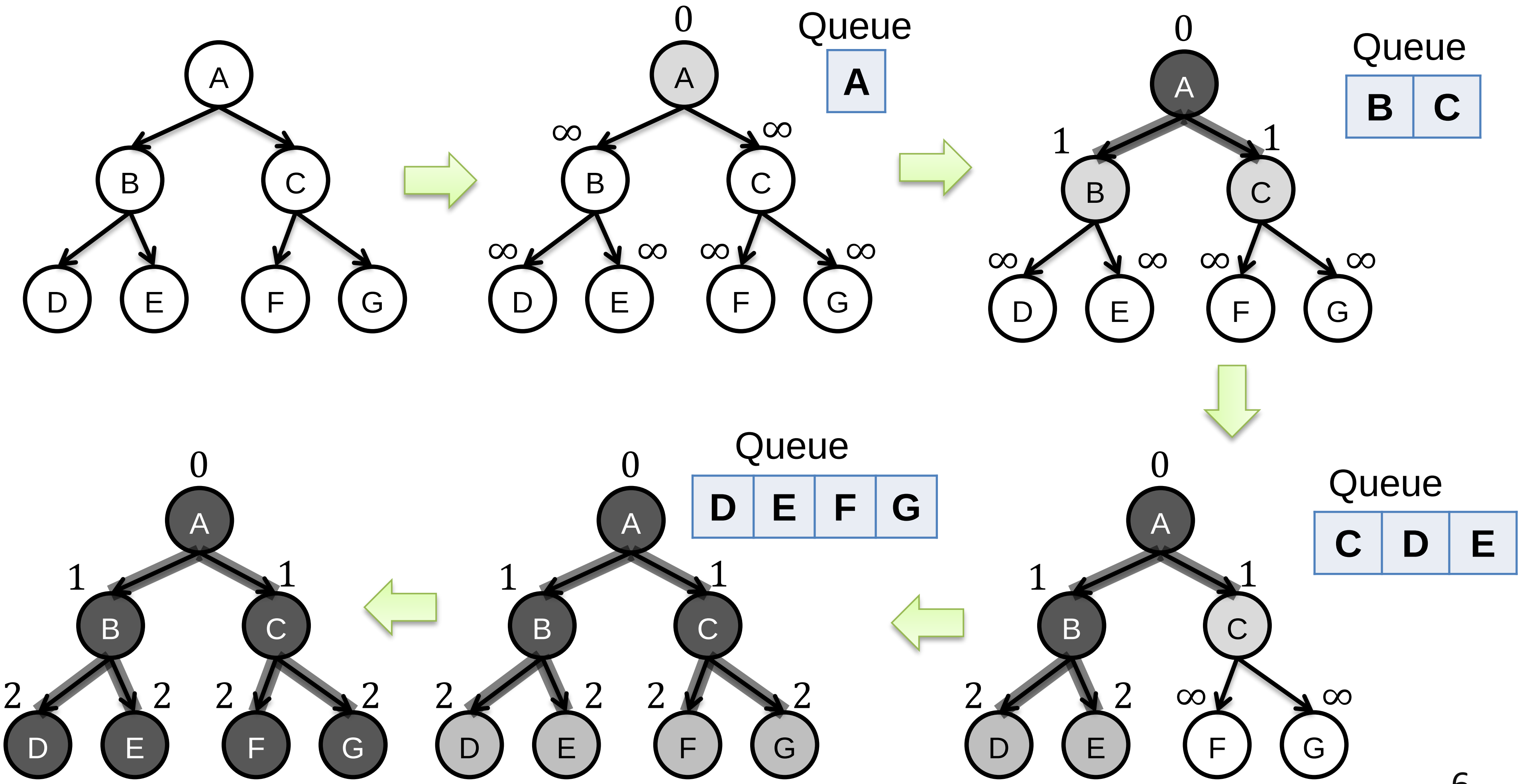


```
BFS(G, s)
for each vertex $u \in G.V - \{s\}$
    $u.color$ = WHITE
    $u.d = \infty$
    $u.\pi$ = NIL
$s.color$ = GRAY
$s.d = 0$
$s.\pi$ = NIL
$Q = \emptyset$
ENQUEUE($Q, s$)
while $Q \neq \emptyset$
    $u$ = DEQUEUE($Q$)
    for each $v \in G.Adj[u]$
        if $v.color$ == WHITE
          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



# Example

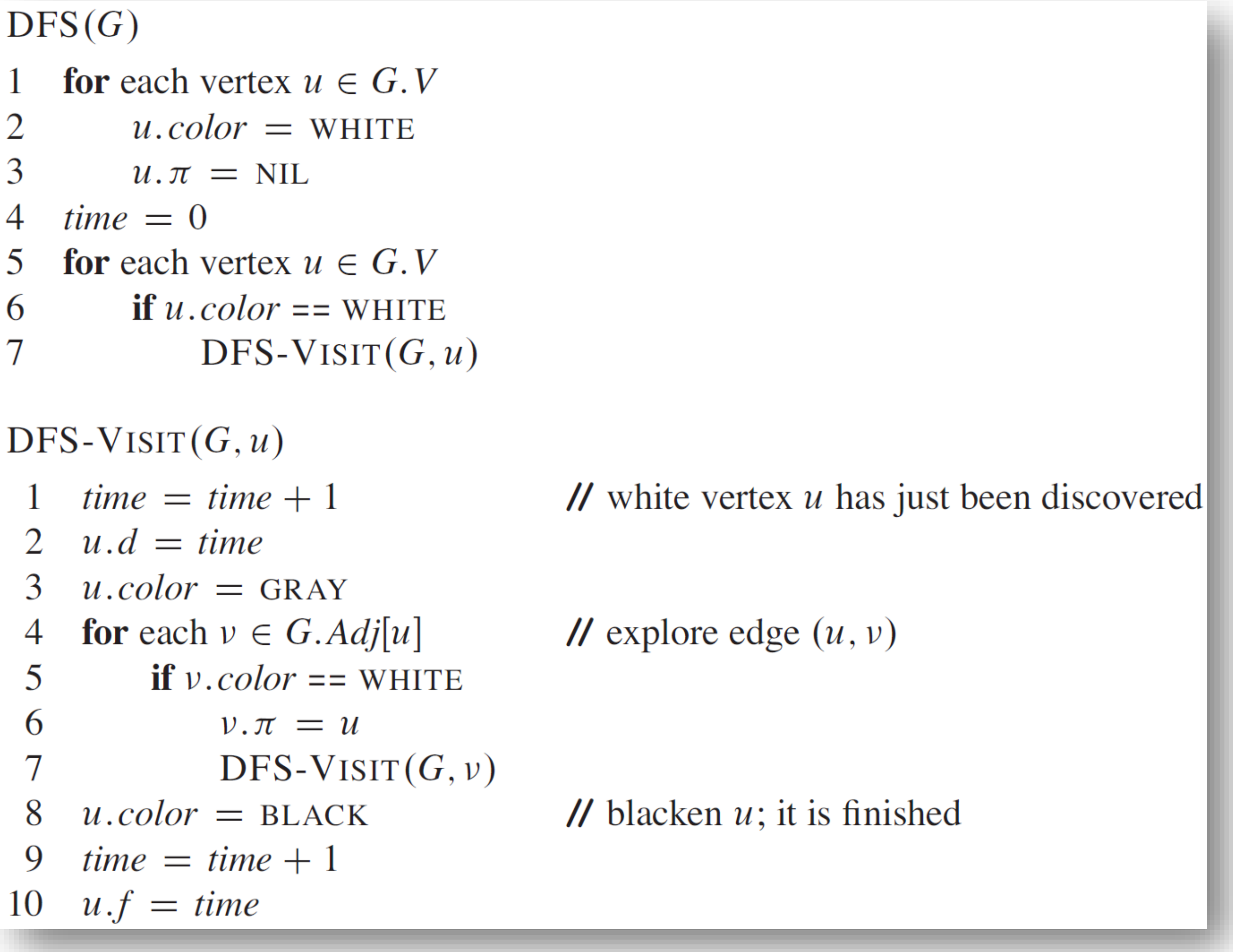
- Given a directed graph $G = (V, E)$, please traverse the graph from node A by using BFS



Traversal sequence: ABCDEFG

# Depth-first Search.

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



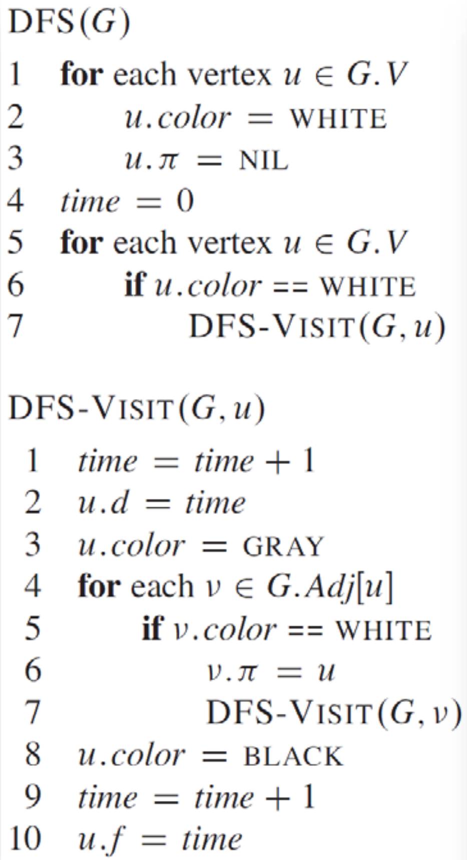
```
DFS($G$)
for each vertex $u \in G.V$
    $u.color = \text{WHITE}$
    $u.\pi = \text{NIL}$
$time = 0$
for each vertex $u \in G.V$
    if $u.color == \text{WHITE}$
        DFS-VISIT($G, u$)

DFS-VISIT($G, u$)
$time = time + 1$                // white vertex $u$ has just been discovered
$u.d = time$
$u.color = \text{GRAY}$
for each $v \in G.Adj[u]$        // explore edge $(u, v)$
    if $v.color == \text{WHITE}$
        $v.\pi = u$
        DFS-VISIT($G, v$)
$u.color = \text{BLACK}$         // blacken $u$; it is finished
$time = time + 1$
$u.f = time$
```

Start time

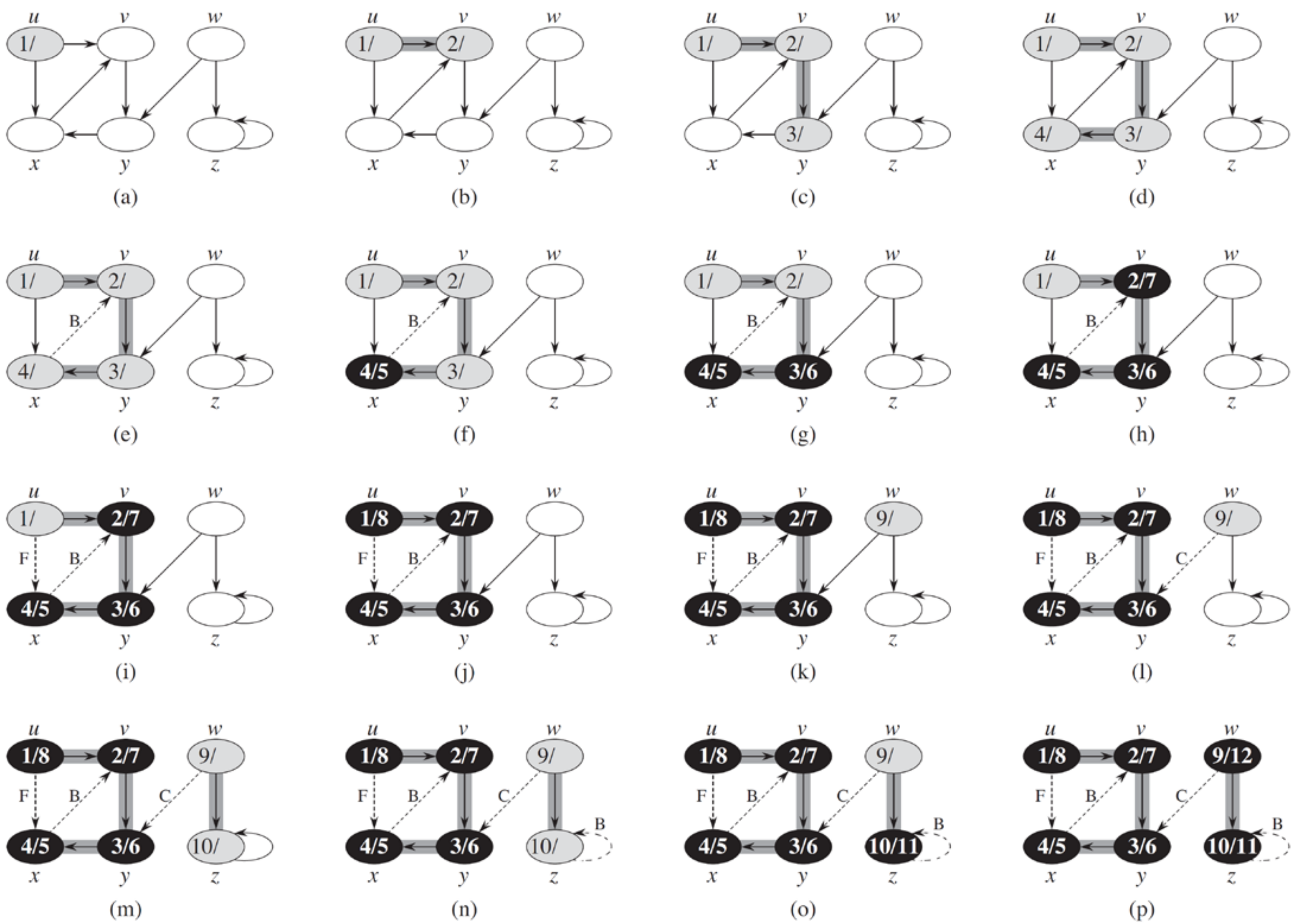
Finish time

# Depth-first Search..

- Based on the “start time,” we can obtain the traversal sequence “uvyxwz”



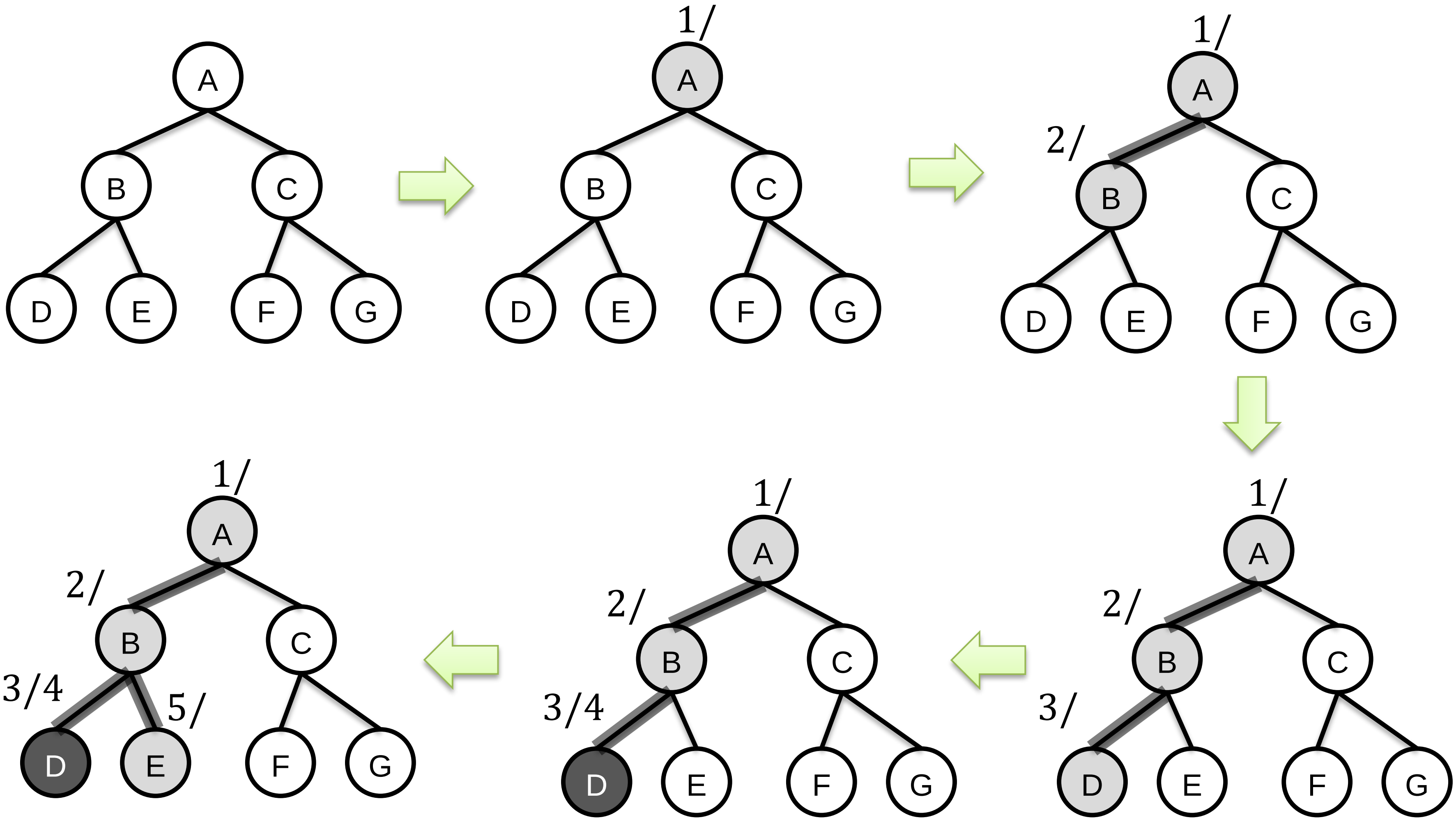
```
DFS($G$)
for each vertex $u \in G.V$
    $u.color$ = WHITE
    $u.\pi$ = NIL
$time = 0$
for each vertex $u \in G.V$
    if $u.color$ == WHITE
        DFS-VISIT($G, u$)

DFS-VISIT($G, u$)
$time = time + 1$
$u.d = time$
$u.color$ = GRAY
for each $v \in G.Adj[u]$
    if $v.color$ == WHITE
        $v.\pi = u$
        DFS-VISIT($G, v$)
$u.color$ = BLACK
$time = time + 1$
$u.f = time$
```



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m) (n) (o) (p)

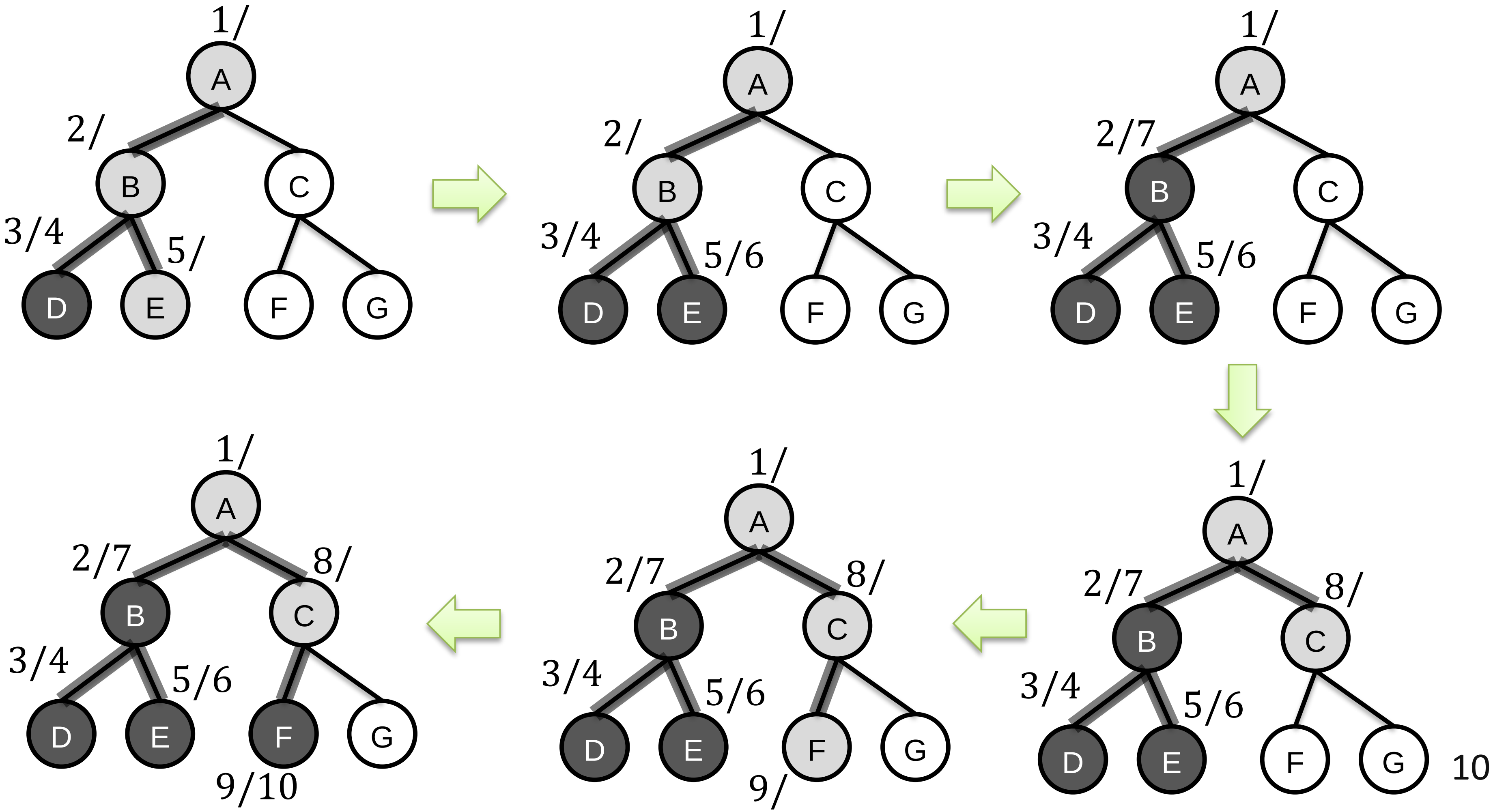
# Example.

- Given an undirected graph $G = (V, E)$, please traverse the graph from A by using DFS



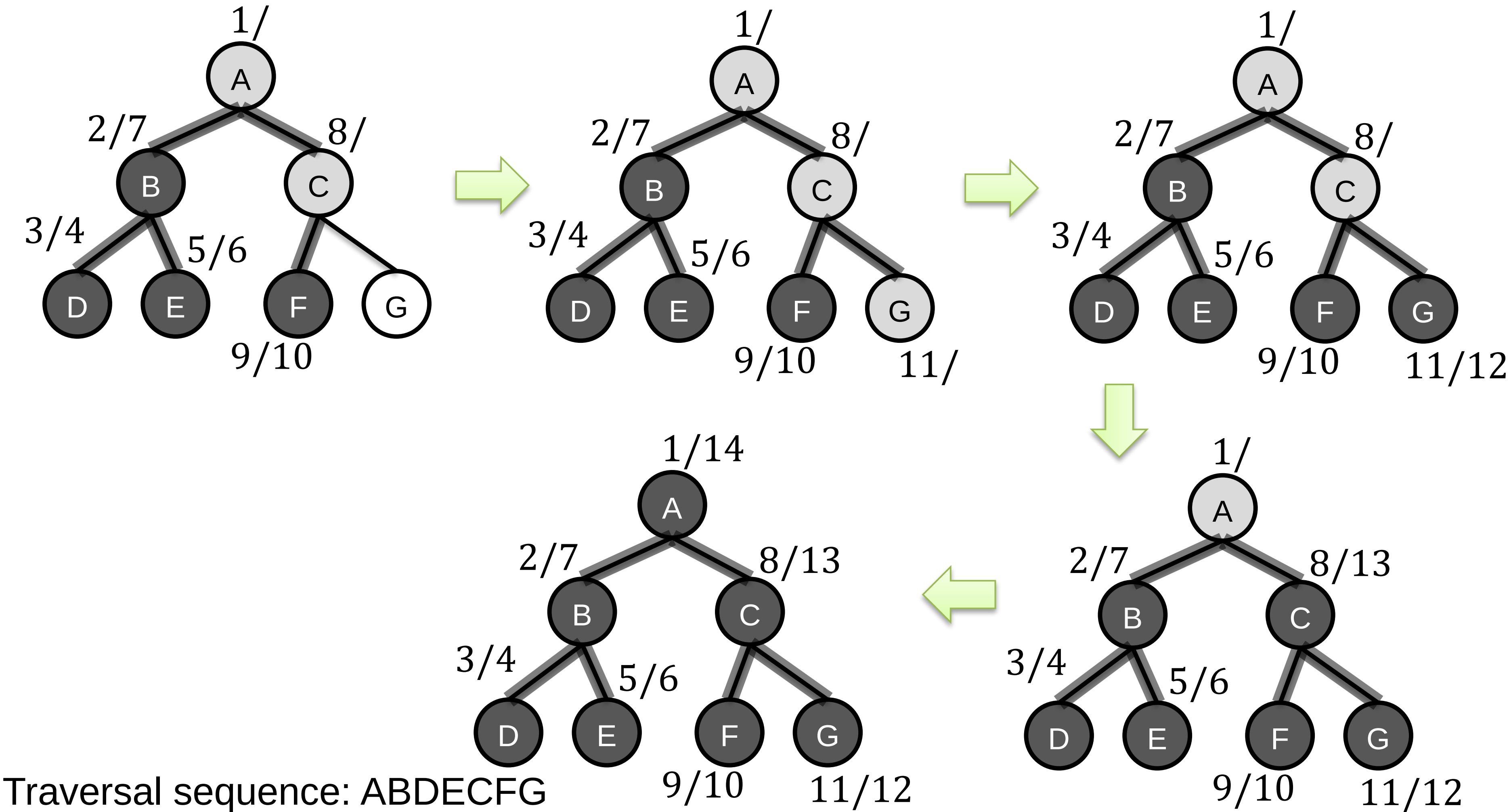
# Example..

- Given an undirected graph $G = (V, E)$, please traverse the graph from A by using DFS



# Example...

- Given an undirected graph $G = (V, E)$, please traverse the graph from A by using DFS



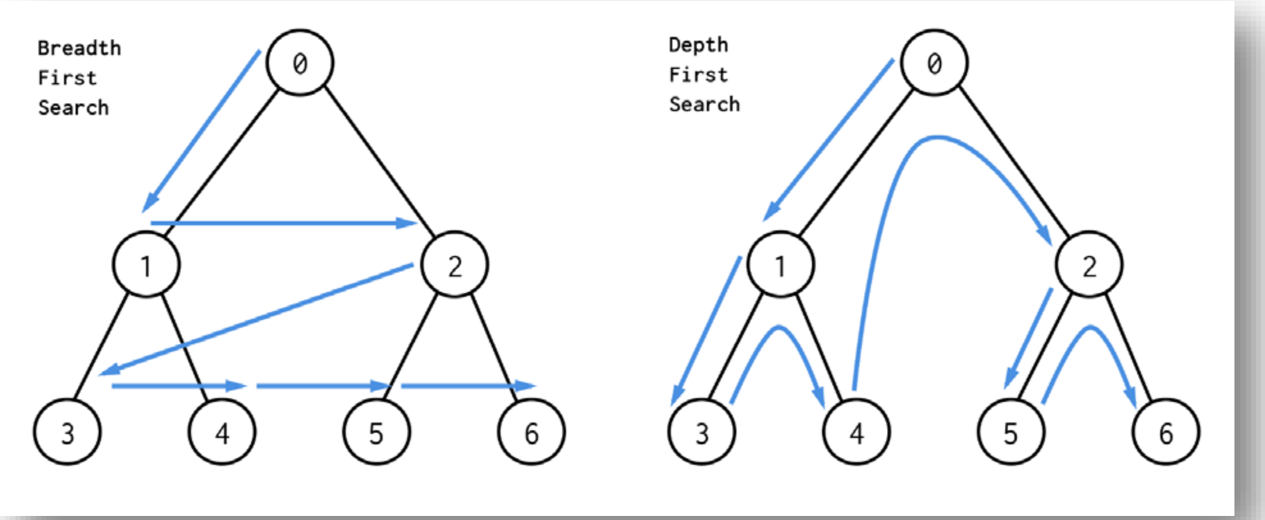
Traversal sequence: ABDECFG

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

# BFS & DFS

- Breadth-first search
  - BFS uses a **queue** as an auxiliary data structure to store nodes for further processing
    - Similar to the level-order of the tree traversal

- Depth-first search
  - DFS uses a **stack** to store nodes for further processing
    - Similar to the pre-order of the tree traversal



# Questions?

kychen@mail.ntust.edu.tw