Ch.22.Elementary Graph Algorithms

Read all sections

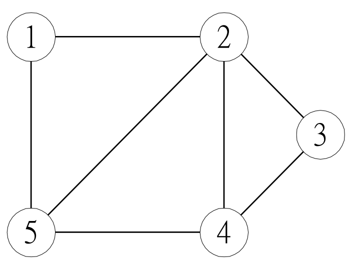
Omit theorems, corollaries, lemmas and their proofs (but learn the results described in the slides)

Conventions

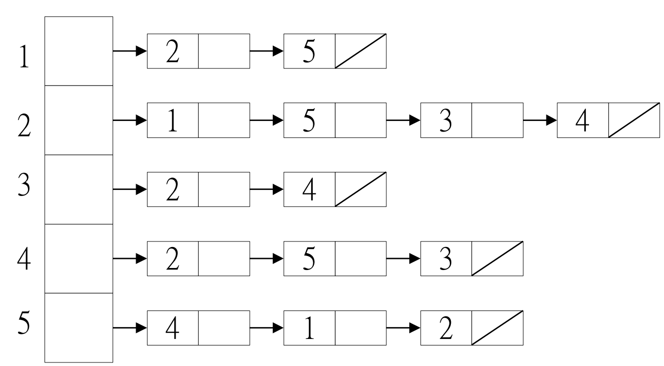
- V – set of vertices or nodes
- E – set of edges
- |V| = # of vertices or nodes = n
- |E| = # of edges = m

22.1 Representations of graphs

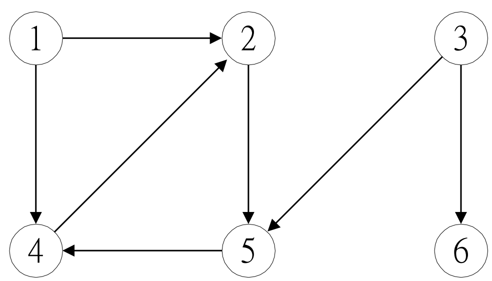
- types: undirected, directed & weighted graphs
  - what is the max # of edges in a n-node undirected graph?
- adjacency matrix representation (for dense graphs)
- adjacency-list representation (for sparse graphs)



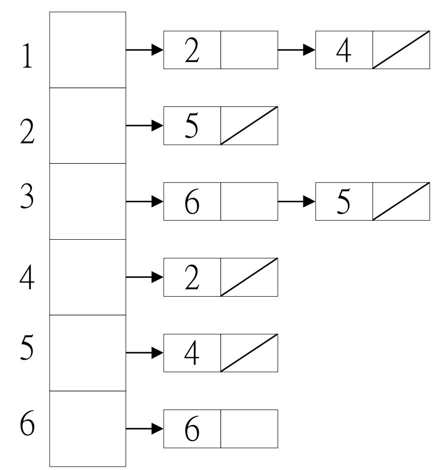
$$\begin{array}{c c}
 & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\
\begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}
\end{array}$$



Understand these examples!



$$
\begin{array}{c|c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
$$



22.2 Breadth-first search

- An algorithm for searching a graph
- Starting from a vertex s, visits every vertex that is reachable from s
- Called breadth-first because it visits all vertices reachable from s in 1 step (by 1 edge) first, then those reachable in 2 steps, and so on.
- Produces a breadth-first tree

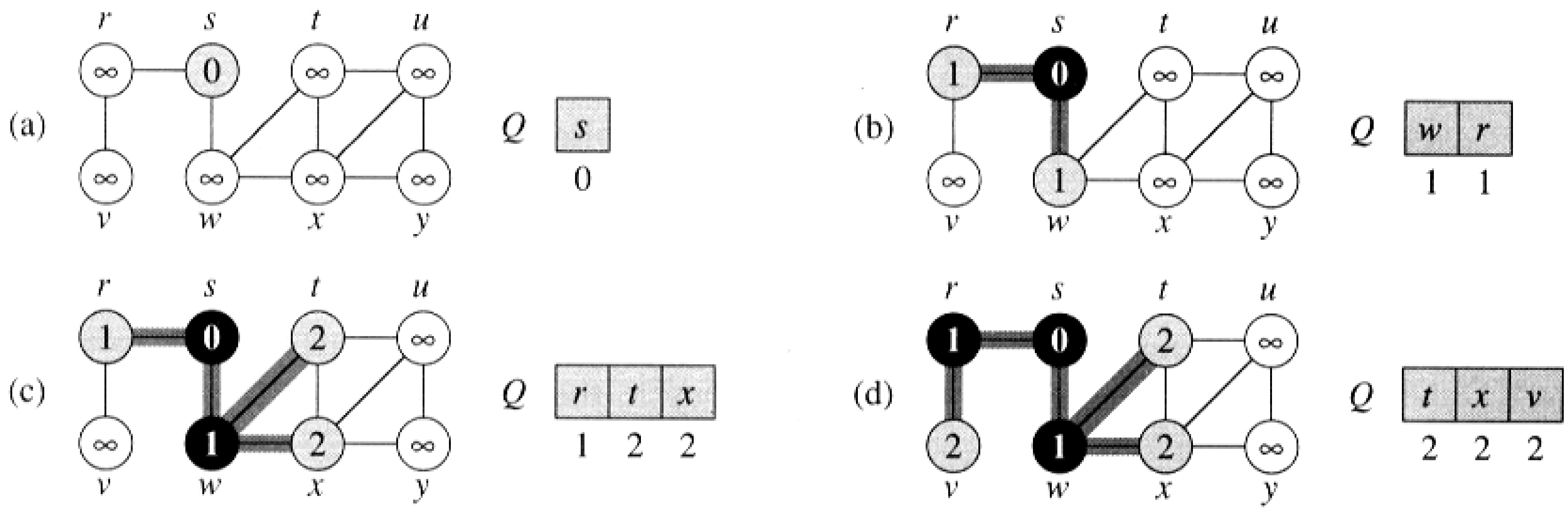
22.2 Breadth-first search

BFS(G, s)

for each vertex $u \in V(G) - \{s\}$
    $color[u] = white$
    $d[u] = \infty$
    $\pi[u] = NIL$
$color[s] = gray$
$d[s] = 0$
$\pi[s] = NIL$
$Q = \{s\}$
while $Q \neq \phi$
    $u = head[Q]$
    for each $v = adj[u]$
        if $color[v] = white$
        then $color[v] = gray$
            $d[v] = d[u] + 1$
            $\pi[v] = u$
            ENQUEUE(Q, v)
    DEQUEUE(Q)
    $color[u] = BLACK$

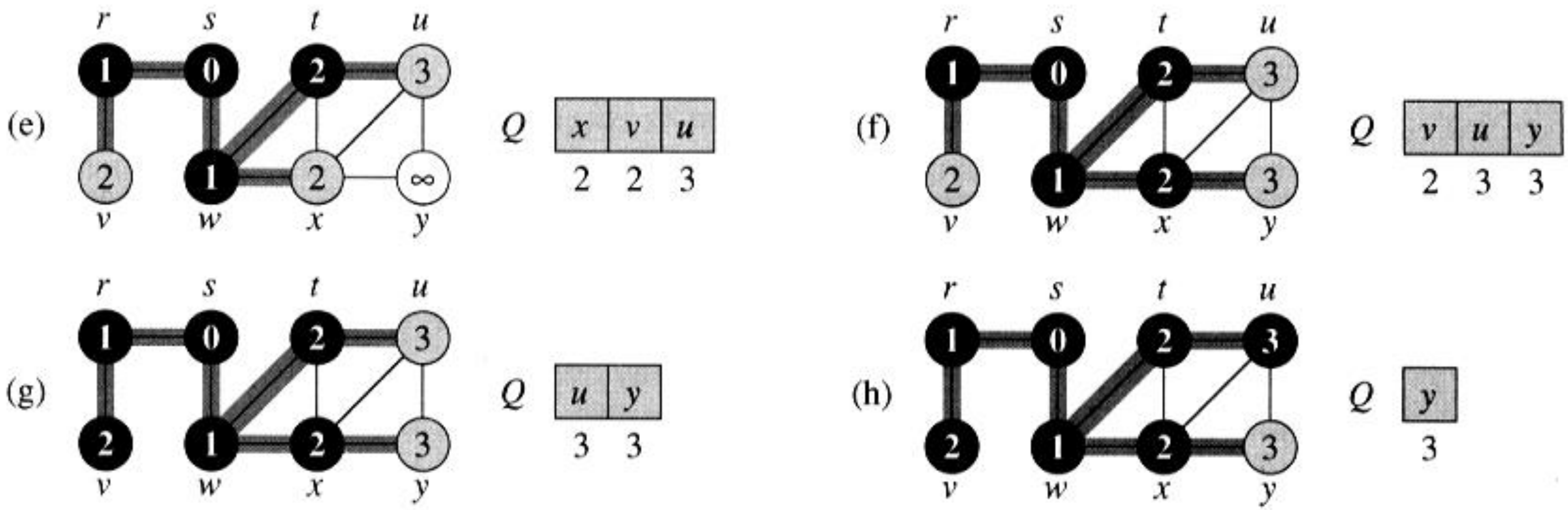
$\boxed{\text{Complexity: } \Theta(|V|+|E|)}$

The operation of BFS

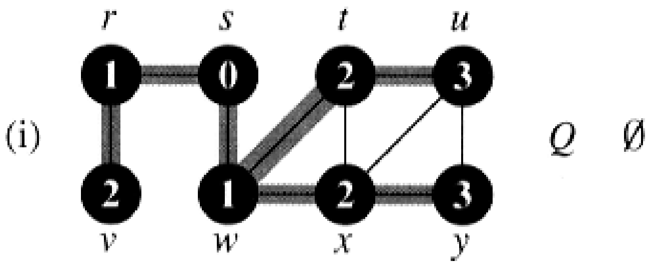


Thinking Assignment: Work out and understand this example yourself

The operation of BFS



The operation of BFS



- Thinking Assignment: Draw the breadth-first tree created by BFS
- Thinking Assignment: What are the distances & shortest paths from s to every other node?

Thinking Assignment

PRINT_PATH(*G*, *source*, *destination*)

if *source* = *destination*

  then print *source*

  else if π[*destination*]=NIL

    then print "no path from" *source* "to" *destination*

    else PRINT-PATH(*G*, *source*, π[*destination*])

      print *destination*

Understand and simulate this recursive algorithm on the previous graph with source=s and destination=y

22.3 Depth-First Search

- Another graph search algorithm
- Depth-first: Go as deep as possible from s, then backtrack to find unvisited nodes
- Builds a depth-first forest (several depth-first trees) in the process
- Places two time stamps on each vertex: d[v] the time at which a vertex is “discovered” and f[v] when it’s processing is finished; Each timestamp is an integer between 1 and 2n

22.3 Depth-First Search

DFS(G)

for each vertex $u \in V[G]$

    $color[u] = white$

    $\pi[u] = NIL$

$time = 0$

for each vertex $u \in V[G]$

    if $color[u] = white$

    then DFS-VISIT(u)

DFS-VISIT(u)

$color[u] = gray$

$d[u] = time = time + 1$

for each $v \in adj[u]$

    if $color[v] = white$

    then $\pi[v] = u$

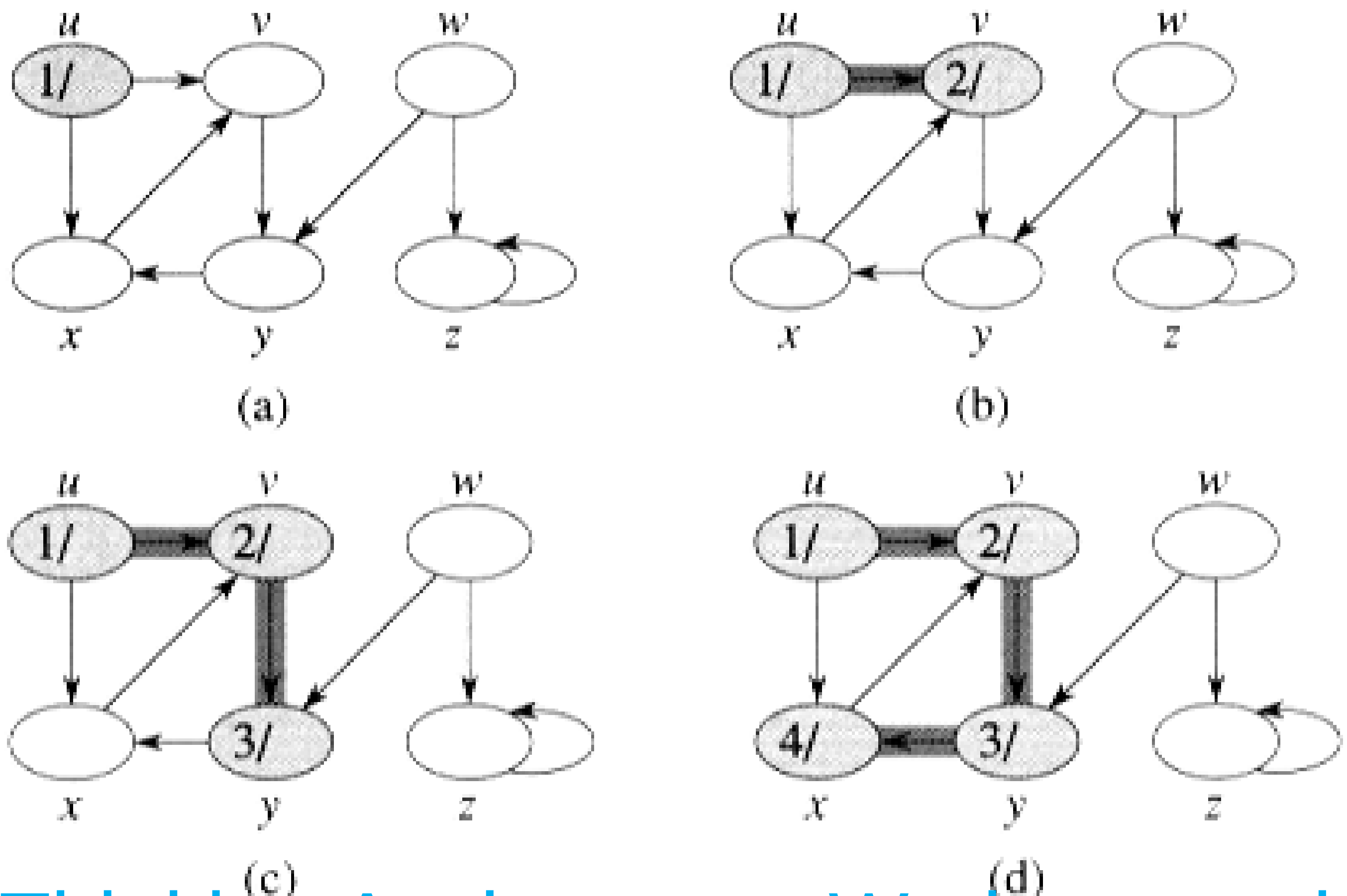
    DFS-VISIT(v)

$color[u] = black$

$f[u] = time = time + 1$

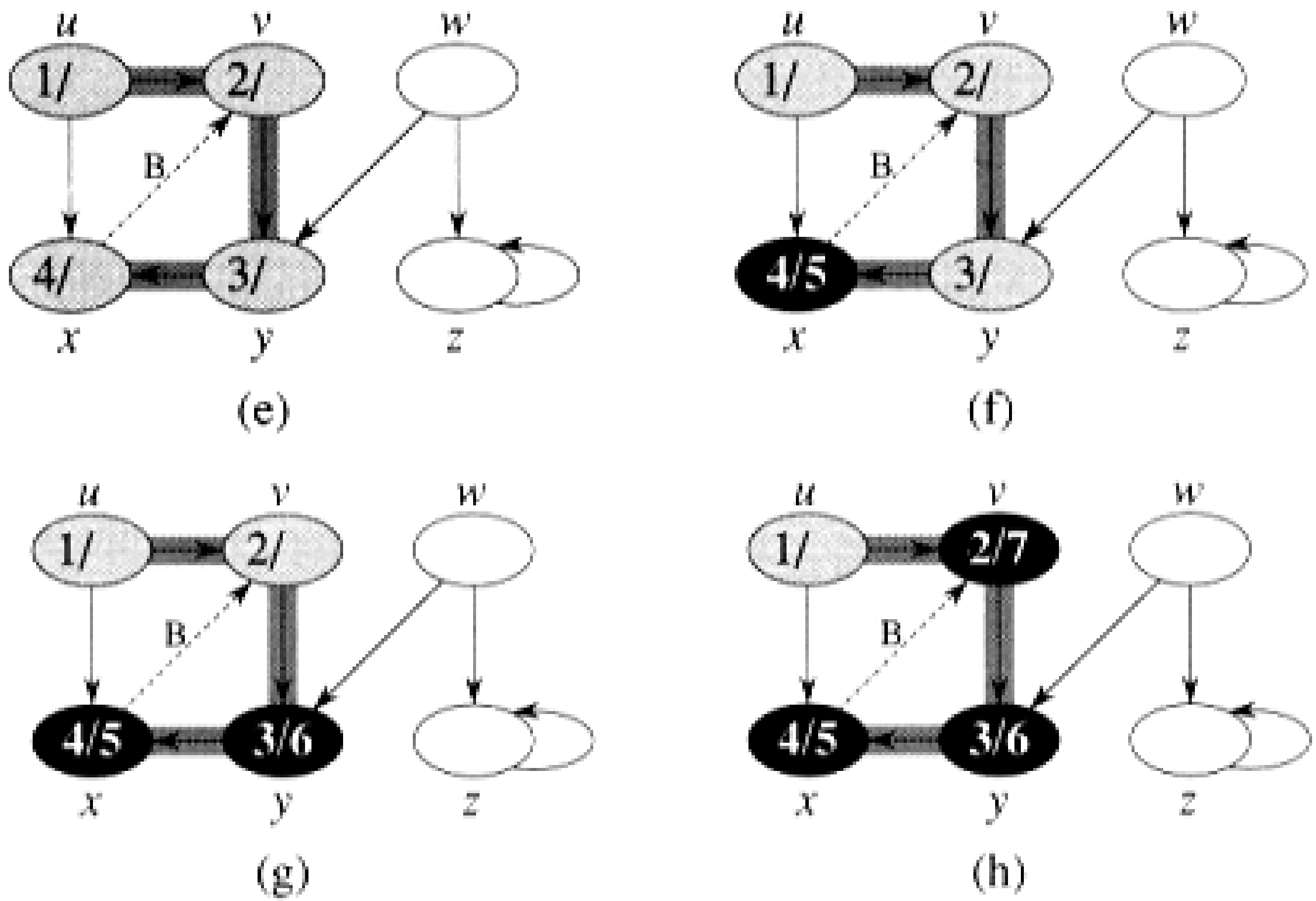
Complexity: $\Theta(|V|+|E|)$

The operation of DFS

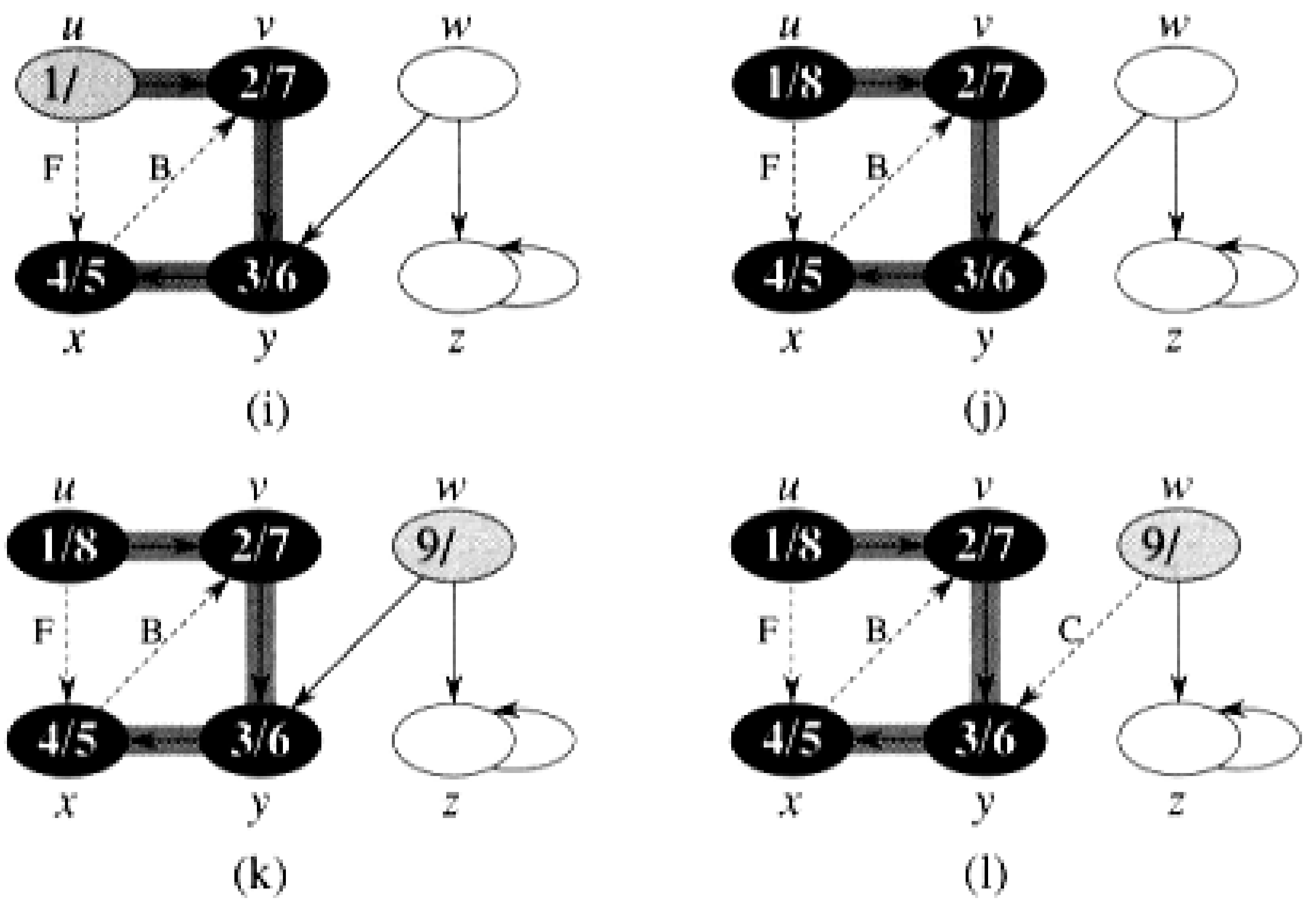


Thinking Assignment: Work out and understand this example yourself

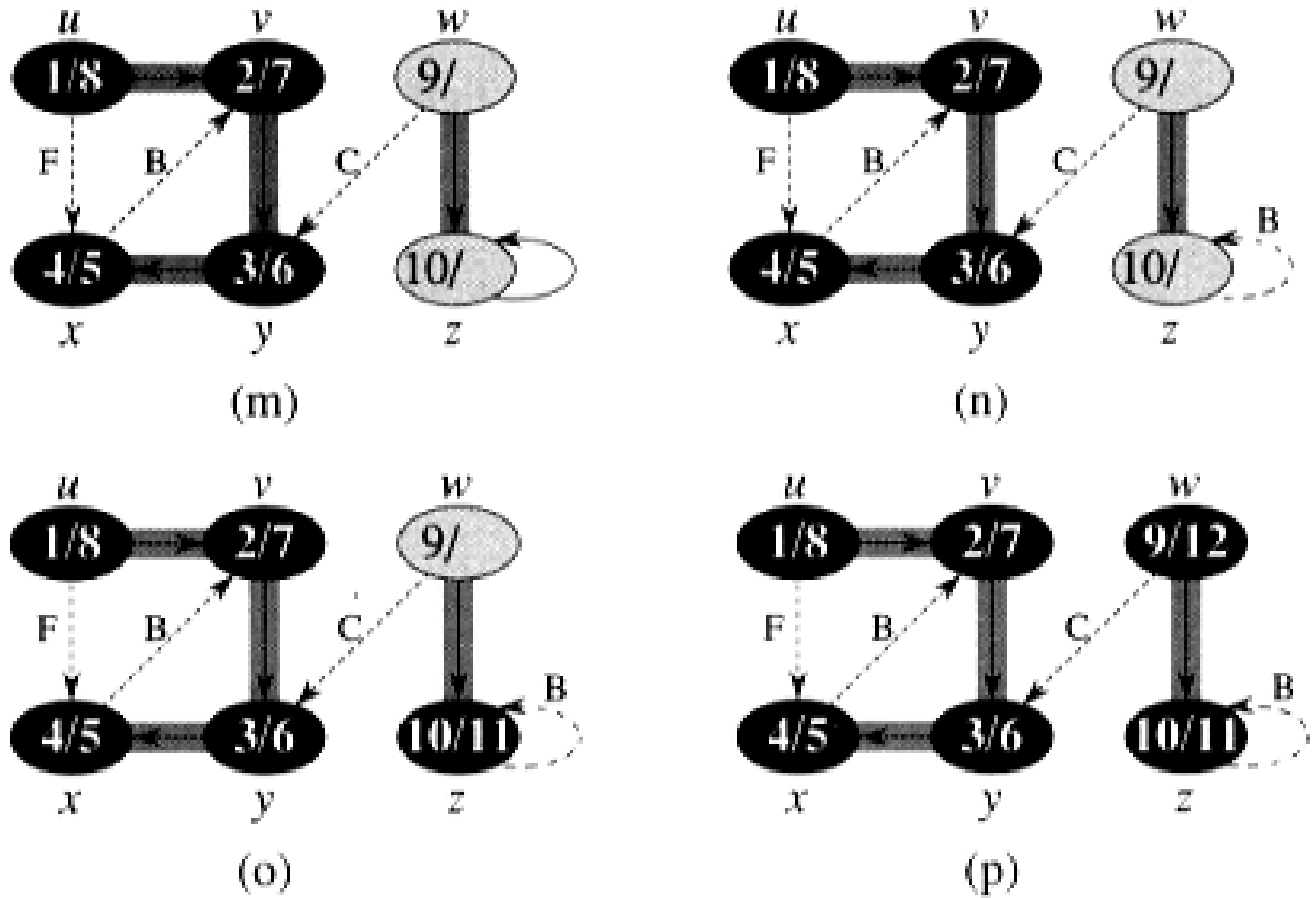
The progress of DFS



The progress of DFS



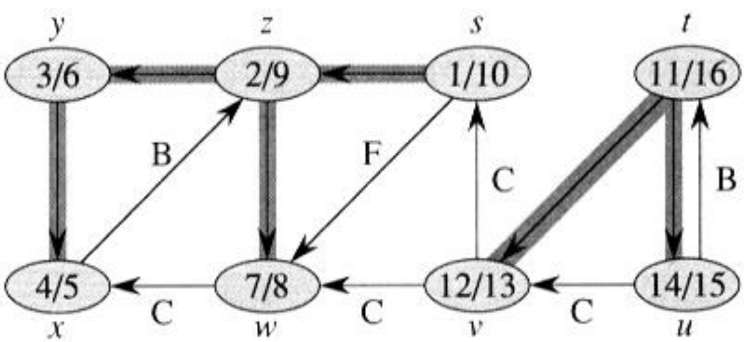
The progress of DFS

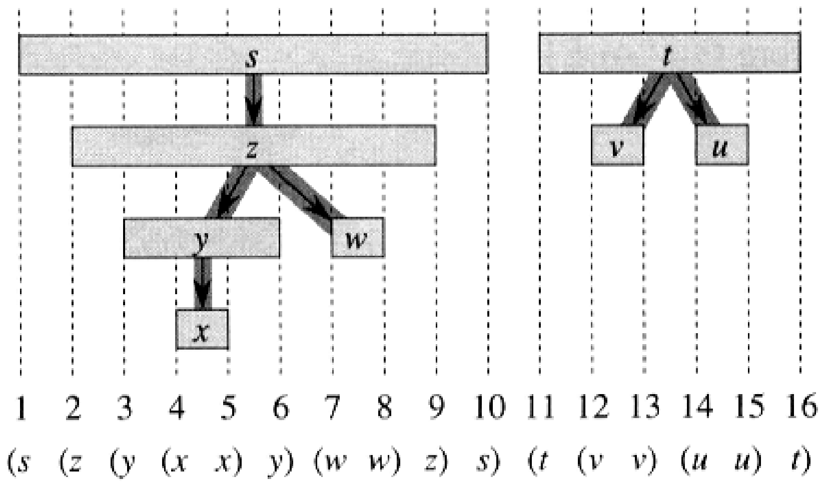


Depth-first forest

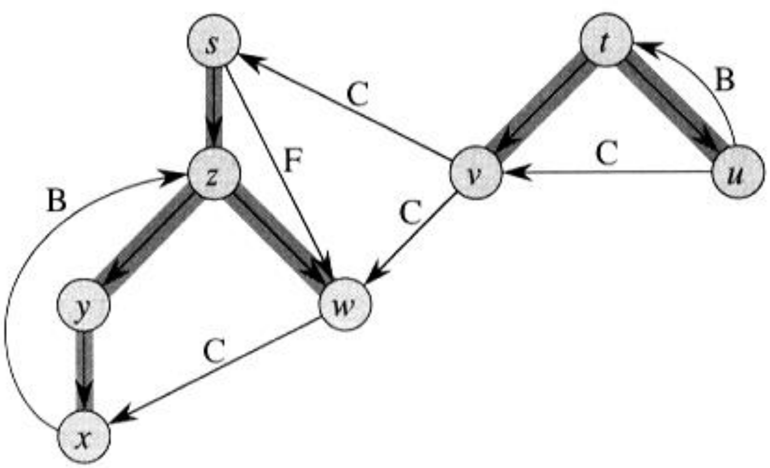
- Depth-first forest consists of trees that represent the edges of the original graph that the DFS algorithm traversed
- These edges are called tree edges
- The remaining edges in the original graph can be classified as back, forward, or cross edges
- DFS of an undirected graph will produce only tree and back edges
- DFS of a directed graph may produce tree, back, forward, or cross edges

Example: Graph, its depth-first forest and other edges





Thinking Assignment: Work out and understand this example yourself



Depth-first search applications

- To check if a graph has a cycle or not: a graph is acyclic if and only if the depth-first forest trees have no back edges
- To check if the graph has one or more nodes whose removal will split the graph into separate pieces: if there are no such nodes (called articulation points) the graph is said to be biconnected.
- To sort the nodes of the graph so that constraints modeled by the graph edges are satisfied: topological sort.

22.4 Topological sort

A topological sort of a directed acyclic graph $G = (V, E)$ is a linear ordering of all its vertices such that if G contains an edge (u, v), then u appears before v in the ordering.

TOPOLOGICAL_SORT(*G*)

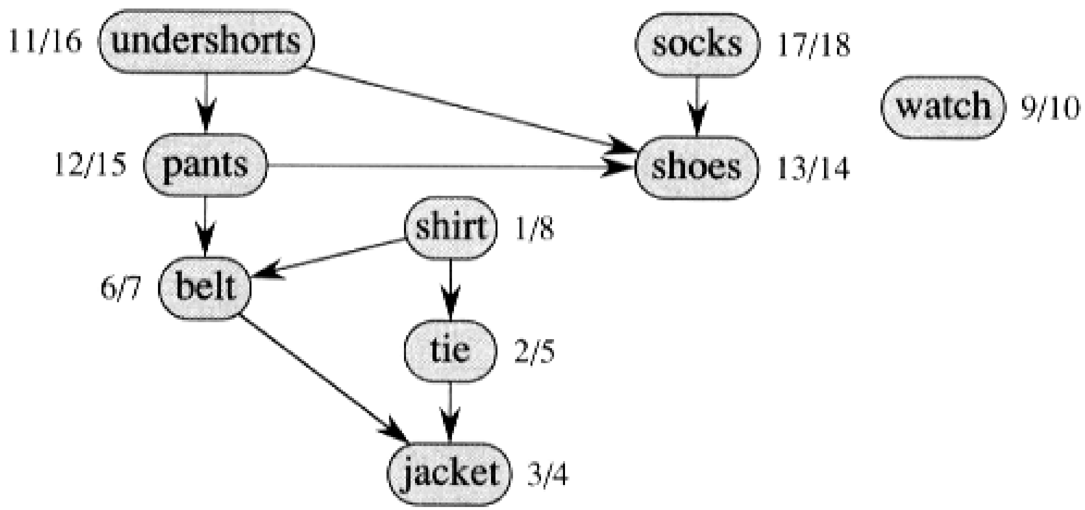
1 call DFS(*G*) to compute finishing time f(*v*) for each vertex *v*.

2 as each vertex is finished, insert it onto the front of a link list.

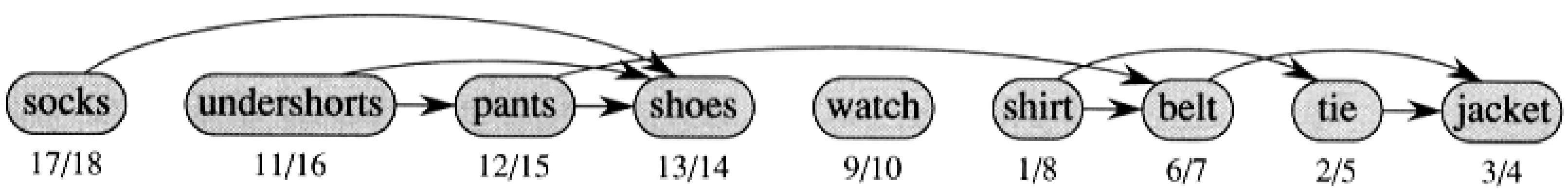
3 return the link list of vertices

$\boxed{\text{Complexity: } \Theta(|V|+|E|)}$

(a)


11/16 undershorts
socks 17/18
watch 9/10
12/15 pants
shoes 13/14
shirt 1/8
6/7 belt
tie 2/5
jacket 3/4

(b)


socks
undershorts
pants
shoes
watch
shirt
belt
tie
jacket
17/18
11/16
12/15
13/14
9/10
1/8
6/7
2/5
3/4

Thinking Assignment

- Come up with a different Topological Algorithm using this strategy:
  - In-degree of a node is the # of edges coming into it; if a node has in-degree zero, then it means that it is not dependent on any other nodes;
    1. therefore, determine the in-degree of all nodes in the graph and store those;
    2. if there is at least one such node (what does it mean if there are no such nodes?), output that as the first node in TO;
    3. then remove all outgoing edges from it by decrementing the in-degrees of all nodes connected by those outgoing edges;
    4. if that makes the in-degree of any node to be zero, output that as the first node in TO (what does it mean if there are no such nodes?);
    5. repeat steps 2-4 until all nodes have been output to the TO