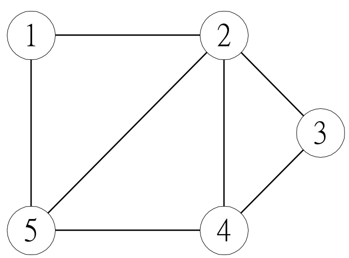
22.Elementary Graph Algorithms

Yu-Shuen Wang, CS, NCTU

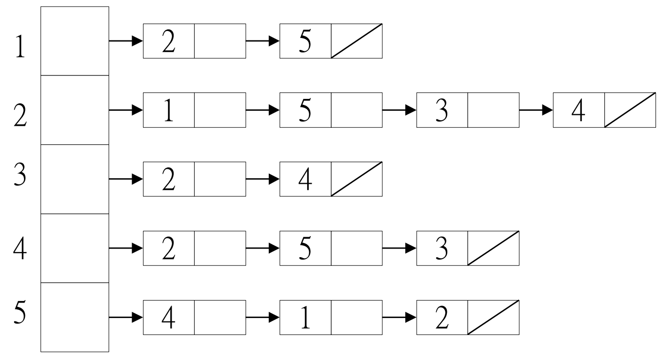
Thanks the images from http://alrightchiu.github.io/SecondRound/mu-lu-yan-suan-fa-yu-zi-liao-jie-gou.html

22.1 Representations of graphs

- adjacency-matrix representation (dense)
- adjacency-list representation (sparse)

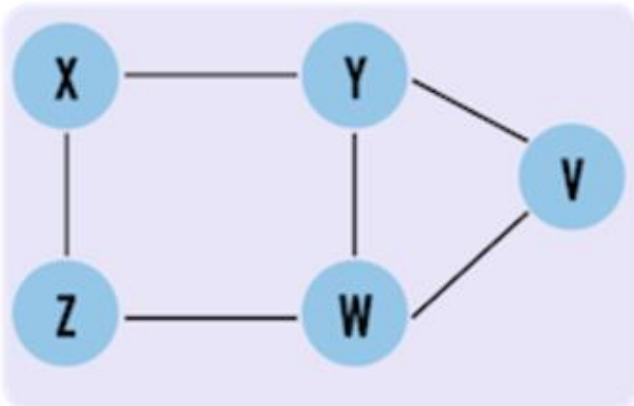


$$
\begin{array}{c c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}
\end{array}
$$



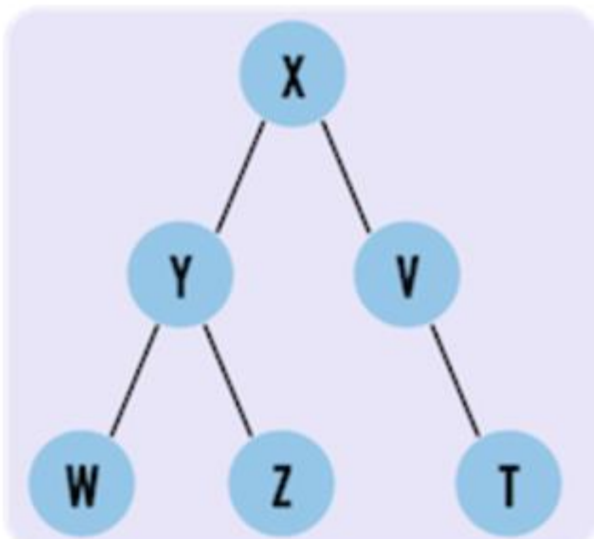
Undirected Graph

G_1



$V(G_1)$: { X, Y, Z, W, V }
$E(G_1)$: { (X, Y), (X, Z), (Y, W), (Y, V), (Z, W), (W, V) }
$|V| = 5$, $|E| = 6$

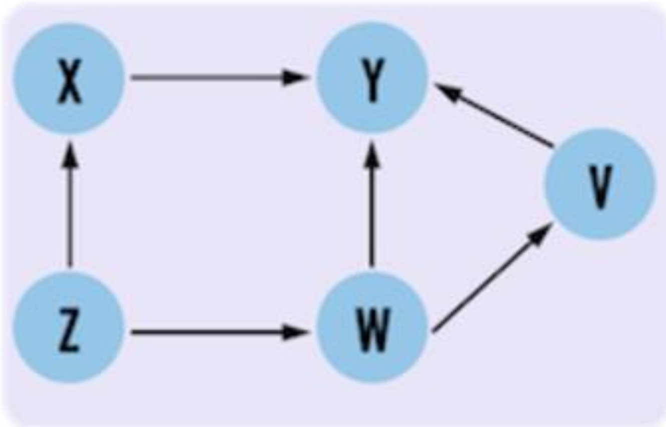
G_2



$V(G_2)$: { X, Y, Z, W, V, T }
$E(G_2)$: { (X, Y), (X, V), (Y, W), (Y, Z), (V, T) }
$|V| = 6$, $|E| = 5$

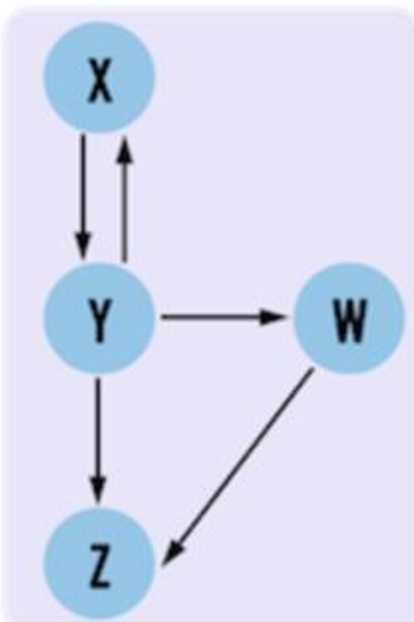
Directed Graph

G_3



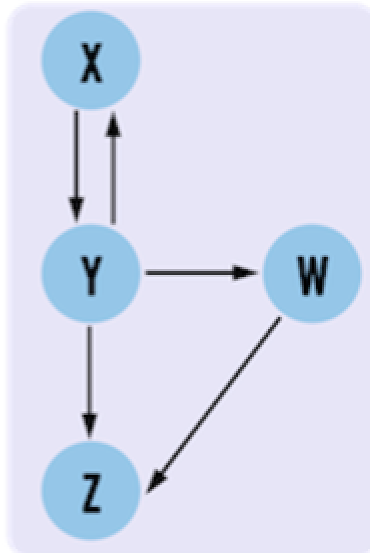
$V(G_3)$: { X, Y, Z, W, V }
$E(G_3)$: { X→Y, W→Y, W→V, Z→X, Z→W, V→Y }
$|V| = 5$, $|E| = 6$

G_4



$V(G_4)$: { X, Y, Z, W }
$E(G_4)$: { X→Y, Y→X, Y→Z, Y→W, W→Z }
$|V| = 4$, $|E| = 5$

Graph G



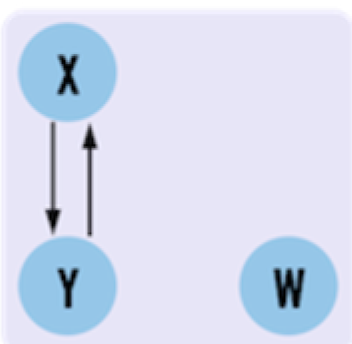
$V(G) : \{X, Y, Z, W\}$

$E(G) : \{X \to Y, Y \to X, Y \to Z, Y \to W, W \to Z\}$

$|V| = 4$, $|E| = 5$

Subgraph of G

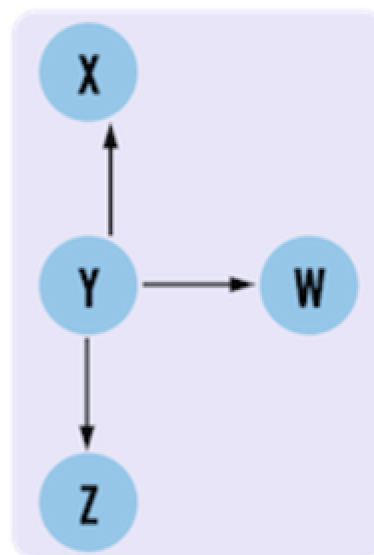
G_1



$V(G_1) : \{X, Y, W\}$

$E(G_1) : \{X \to Y, Y \to X\}$

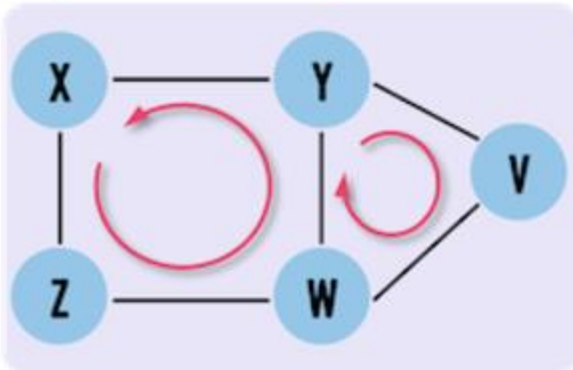
G_2



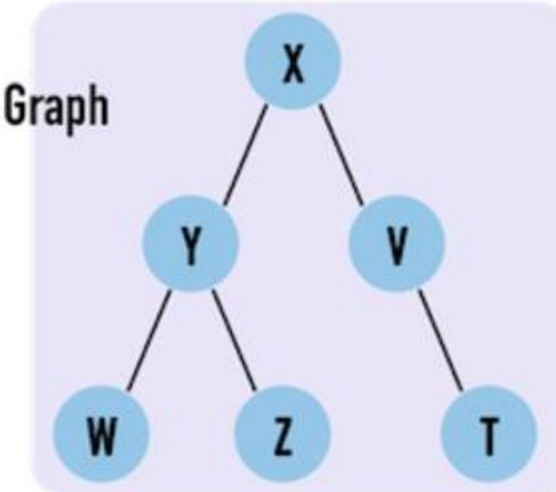
$V(G_2) : \{X, Y, Z, W\}$

$E(G_2) : \{Y \to X, Y \to Z, Y \to W\}$

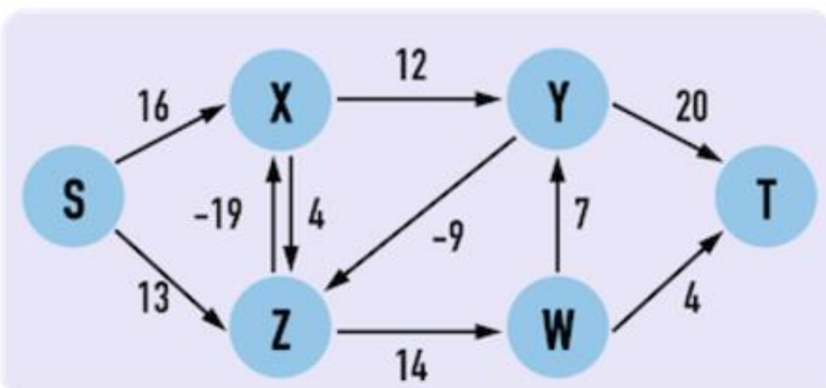
Cycle: Y→V→W→Y

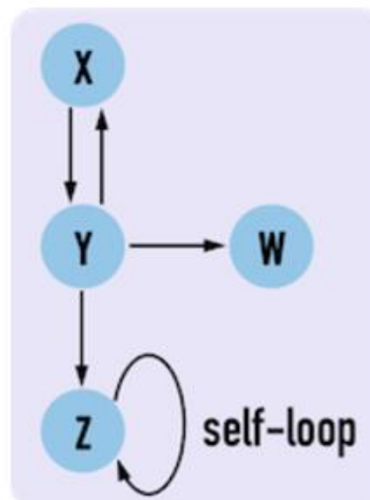


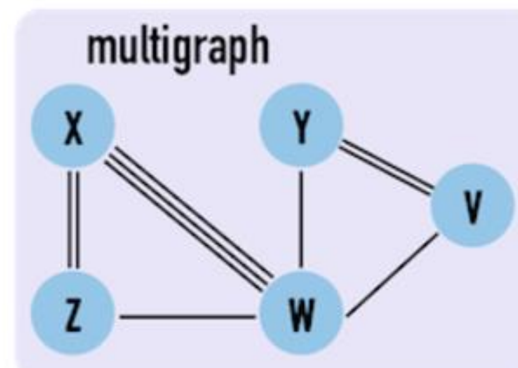
Tree:
Acyclic Graph



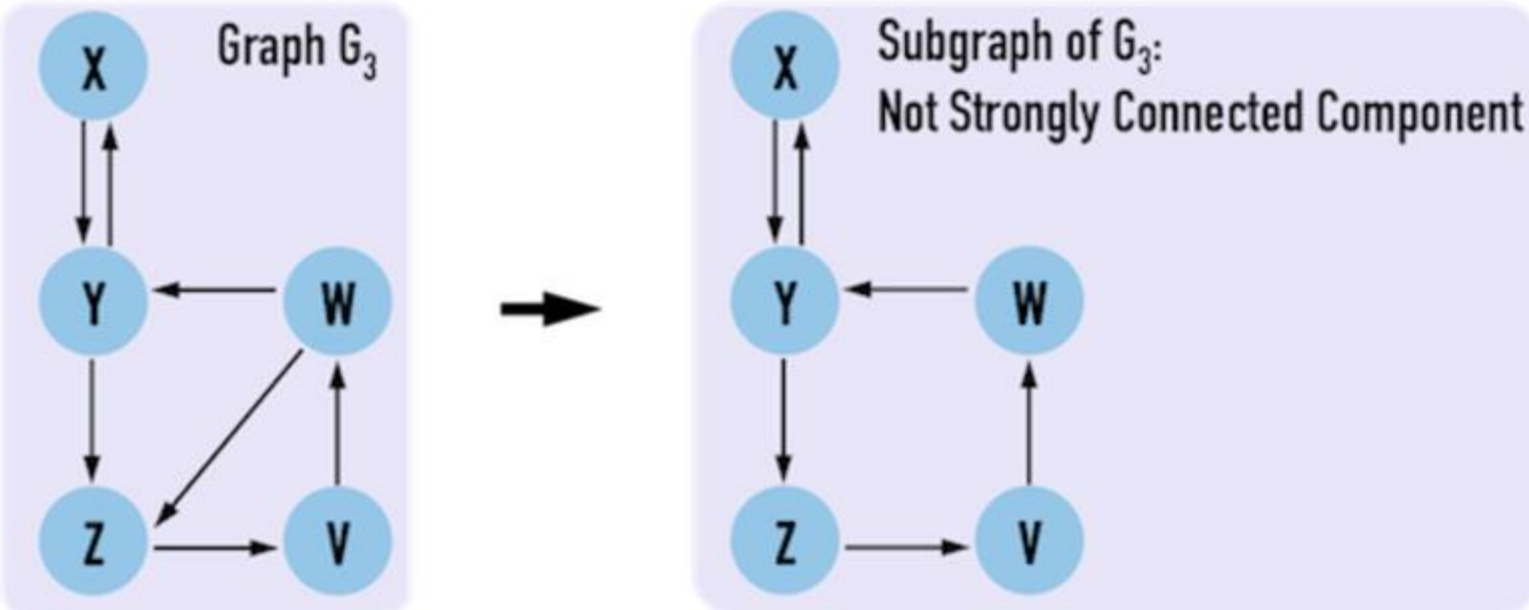
Weighted



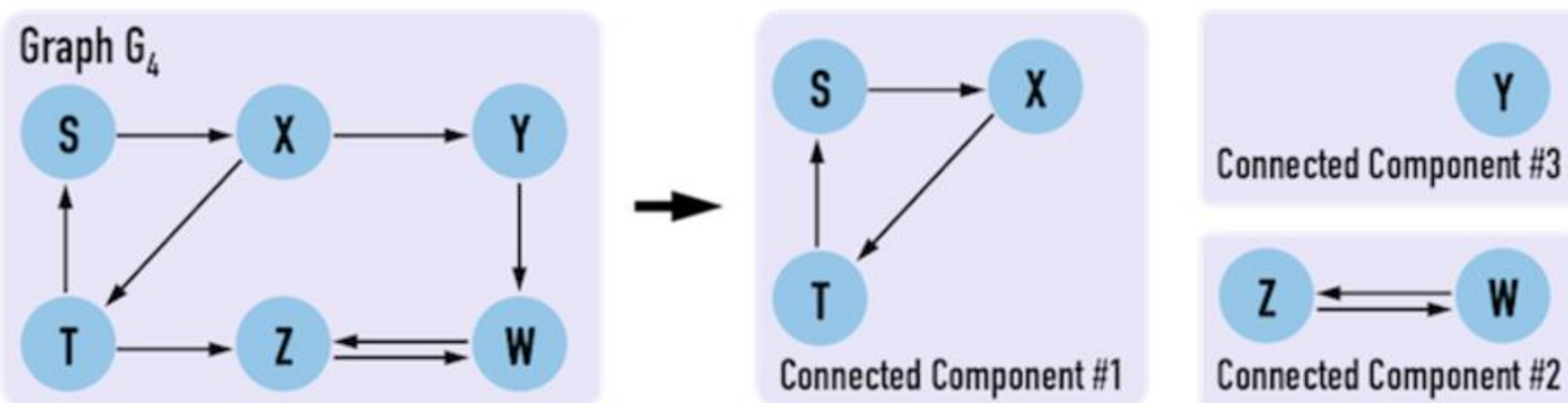




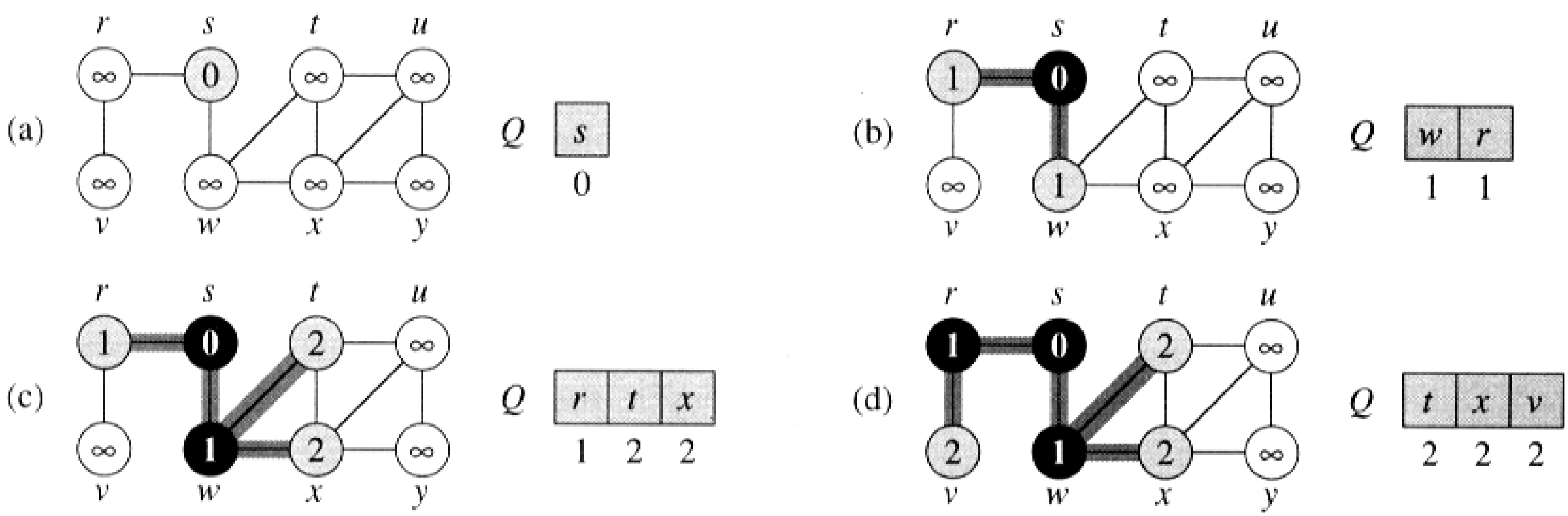
Directed Graph: Strongly Connected



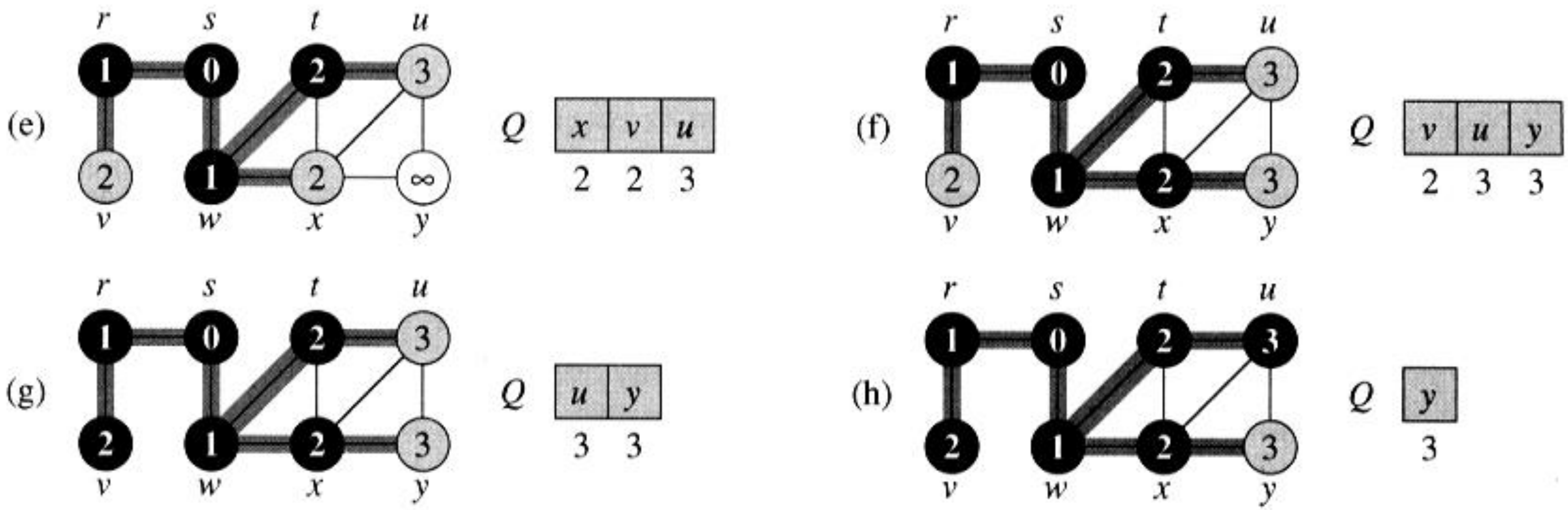
Directed Graph: Not Strongly Connected



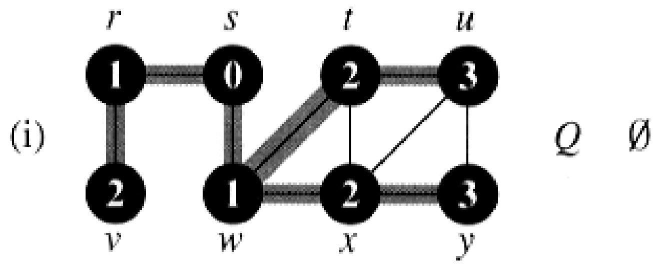
The operation of BFS



The operation of BFS



The operation of BFS



22.2 Breadth-first search

```
BFS(G,s)
    for each vertex u ∈ V[G] - {s}
        do color[u] ← WHITE
            d[u] ← ∞
            π[u] ← NIL
    color[s] ← GRAY
    d[s] ← 0
    π[s] ← NIL
    Q ← {s}
    while Q
    do u ← head[Q]
        for each v ∈ Adj[u]
            do if color[v] = WHITE
                then color[v] ← GRAY
                    d[v] ← d[u] + 1
                    π[v] ← u
                    ENQUEUE(Q,v)
        DEQUEUE(Q)
        color[u] ← BLACK
```

$\boxed{\text{Analysis: } O(V+E)}$

Shortest paths

$\delta(s,v)$: shortest path from s to v

Lemma 22.1. Let $G = (V, E)$ be a directed or undirected graph, and let $s \in V$ be an arbitrary vertex. Then for any edge $(u,v) \in E$, $\delta(s,v) \leq \delta(s,u) + 1$.

Lemma 22.2. Let $G = (V, E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source $s \in V$. Then upon termination, for each vertex $v \in V$, the value $d[v]$ computed by BFS satisfies $d[v] \ge \delta(s,v)$.

Proof. (Induction on the number of times a vertex is placed in the queue)

Lemma 22.3. Suppose that during the execution of BFS on a graph $G = (V, E)$, the queue Q contains the vertices $< v_1, v_2, \ldots, v_r >$, where v_1 is the head of Q and v_r is the tail. Then $d[v_r] \le d[v_1] + 1$ and $d[v_i] \le d[v_{i+1}]$ for $i=1,2,\ldots,r-1$.

Proof. (induction on the number of queue operations)

Corollary 22.4. Suppose vertices v_i and v_j are enqueued during the execution of BFS, and that v_i is enqueued before v_j is enqueued. ***Then*** $d[v_i] \le d[v_j]$ ***at the time that*** v_j ***is enqueued.***

proof Immediate form Lemma 22.3 and the property that each vertex receives a finite *d* value at most once during the course of BFS.

We can now prove that breadth-first search correctly finds shortest-path distances.

Theorem 22.5

Let $G = (V, E)$ be a directed or undirected graph, and suppose that BFS is run on G from a given source $s \in V$. Then, during its execution, BFS discovers every vertex $v \in V$ that is reachable from the source, and upon termination $d[v] = \delta(s,v)$ for all $v \in V$ Moreover, for any $v \neq s$ that is reachable from s, one of the shortest paths from s to v is the shortest path from s to $\pi[v]$ followed by the edge $(\pi[v],v)$.

Proof. By induction.

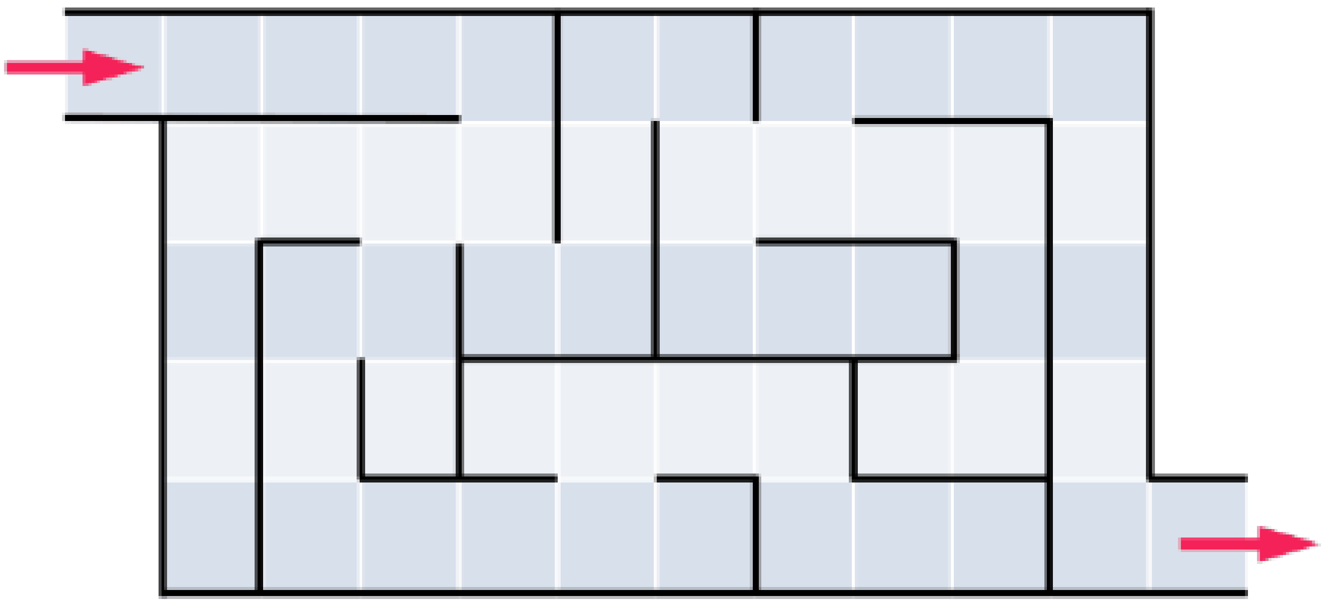
For a graph $G = (V, E)$ with source s, we define the predecessor subgraph of G as $G_\pi = (V_\pi, E_\pi)$ where $V_\pi = \{v \in V \mid \pi[v] \neq NIL\} \cup \{s\}$, and $E_\pi = \{(\pi[v], v) \in E \mid v \in V_\pi - \{s\}\}$. The edges in E_π are called *tree edges*.

Lemma 22.6. When applied to a directed or undirected graph $G = (V, E)$ procedure BFS constructs π so that the predecessor subgraph $G_\pi = (V_\pi, E_\pi)$ is a breadth-first tree.

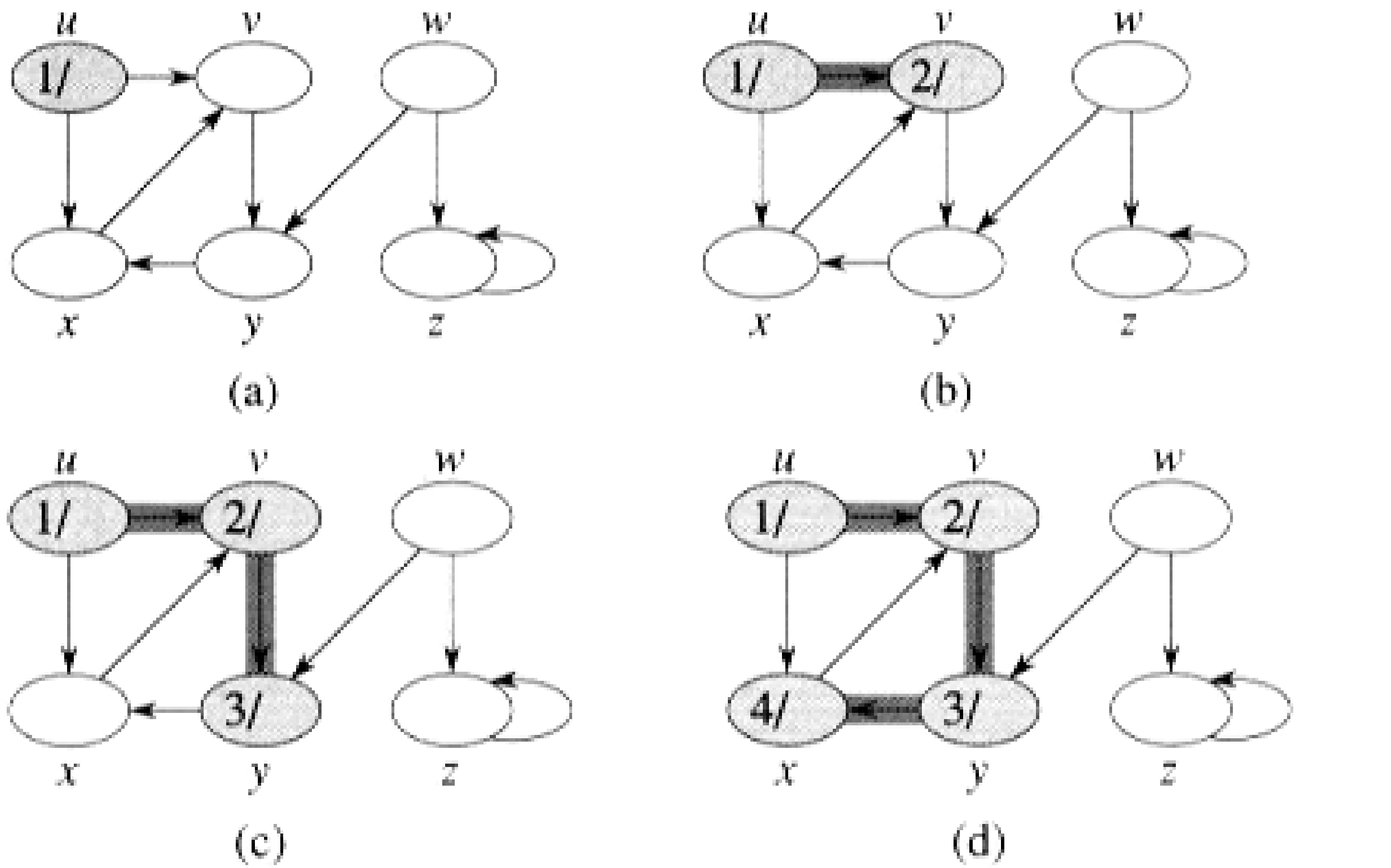
```
PRINT_PATH(G, s, v )
if  v = s
      then print s
      else if  π[v]=NIL
            then print “no path from” s “to” v “exist”
            else PRINT-PATH(G, s, π[v])
                 print v
```

Depth first search

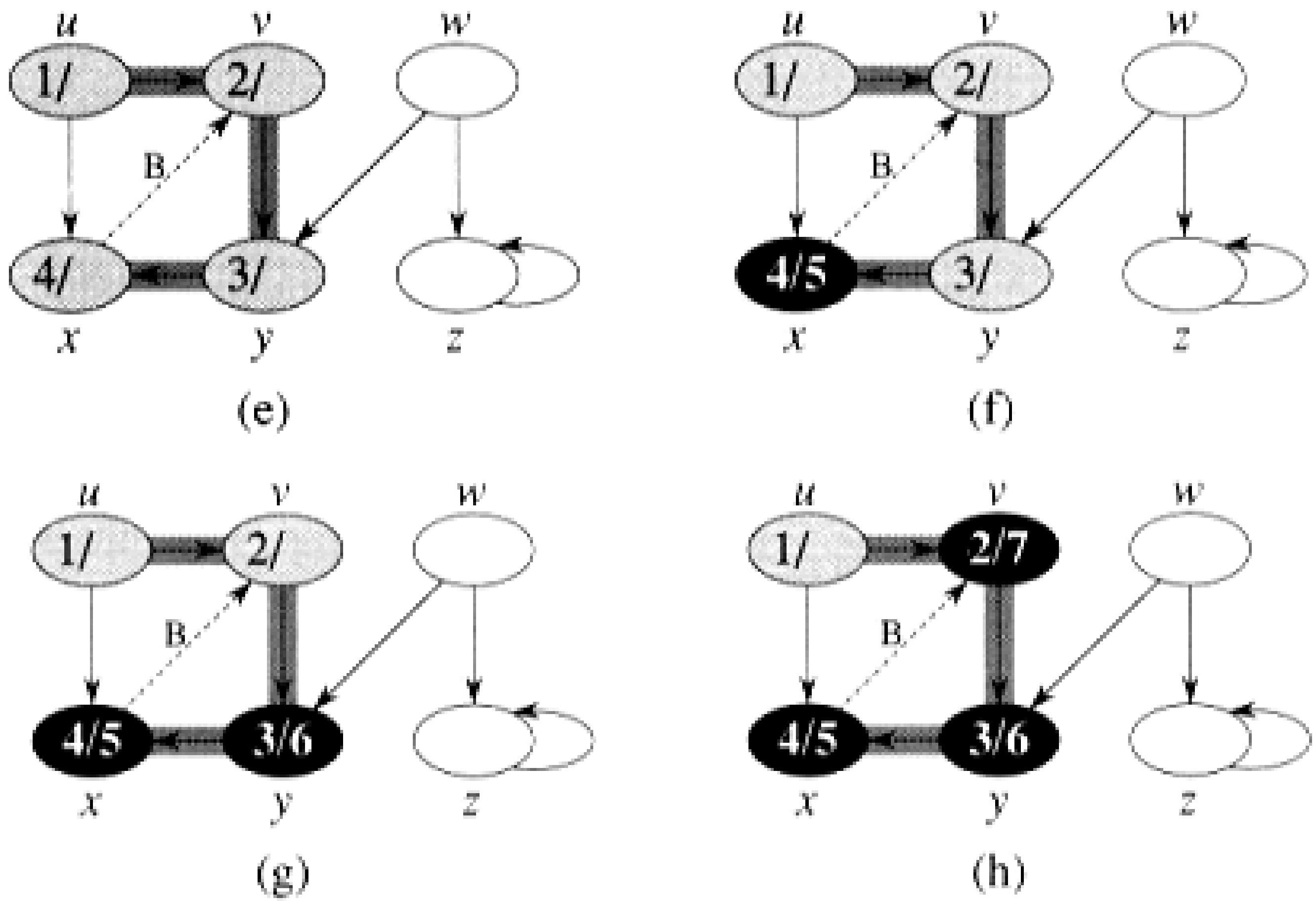
Maze Problem



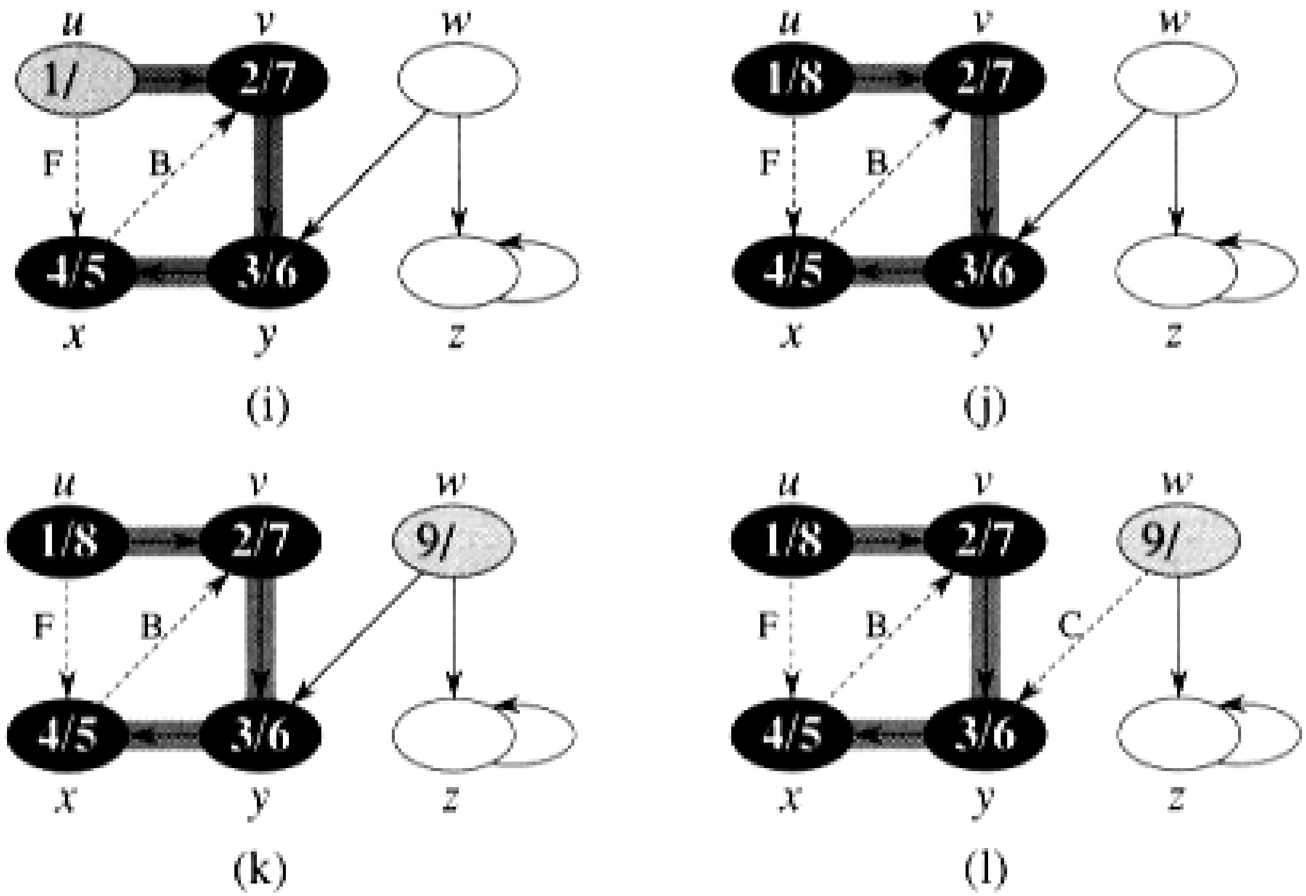
The progress of DFS



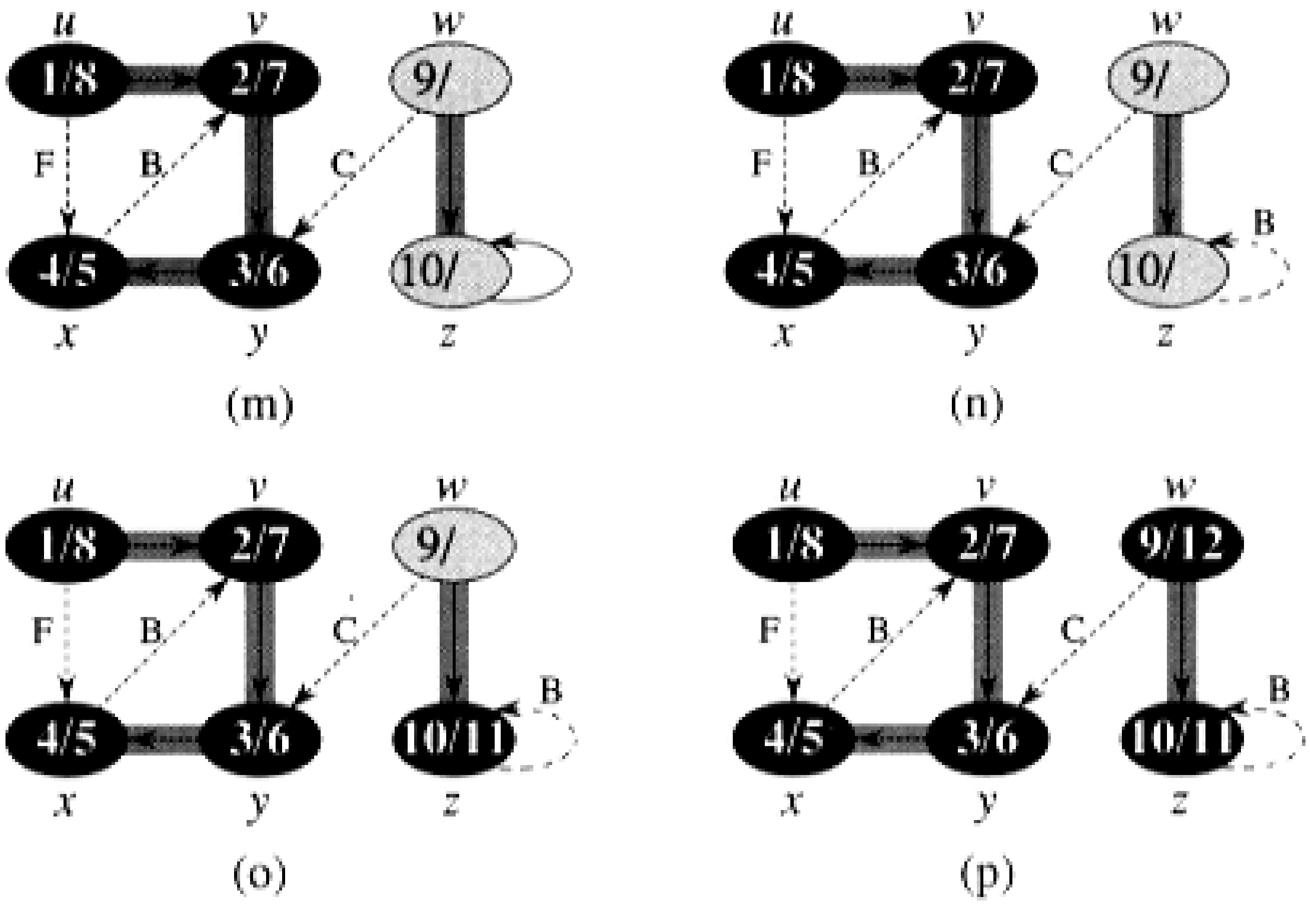
The progress of DFS



The progress of DFS



The progress of DFS



22.3 Depth-First Search

DFS(G)

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

    do $color[u] \leftarrow white$

        $\pi[v] \leftarrow NIL$

$time \leftarrow 0$

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

    do if $color[u] = white$

        then DFS-VISIT(u)

DFS-VISIT(u)

$color[u] = gray$

$d[u] \leftarrow time \leftarrow time + 1$

for each $v \in adj[u]$

    do if $color[u] = white$

        then $\pi[v] \leftarrow u$

            DFS-VISIT(v)

$color[u] = black$

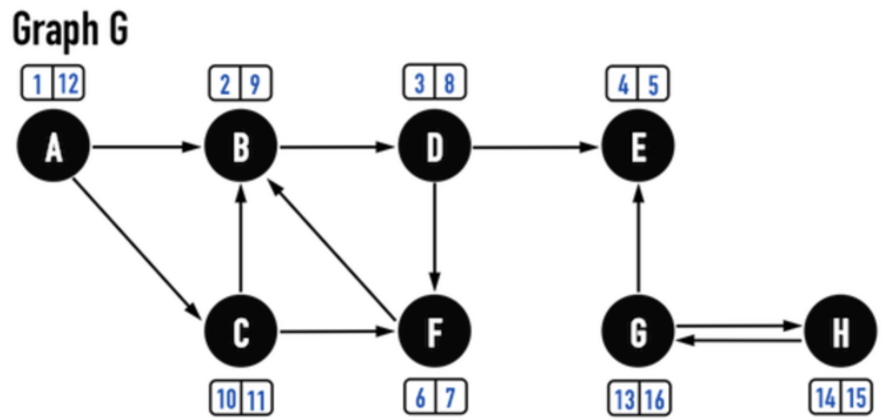
$f[u] \leftarrow time \leftarrow time + 1$

predecessor subgraph:

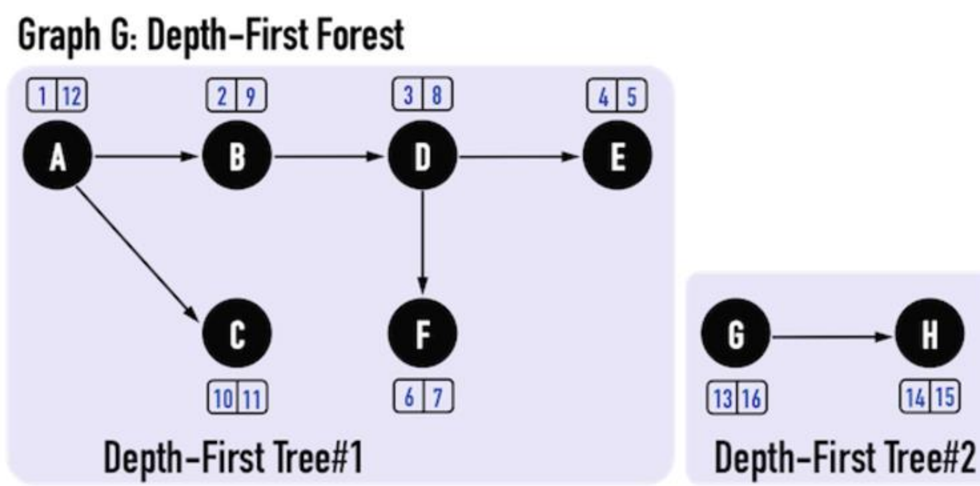
depth-first forest,   depth-first tree

Time stamps: $d(u)$ discovered

$f(u)$ finished



Complexity: $O(V+E)$



Properties of depth-first search

Theorem 22.6. (Parenthesis theorem)

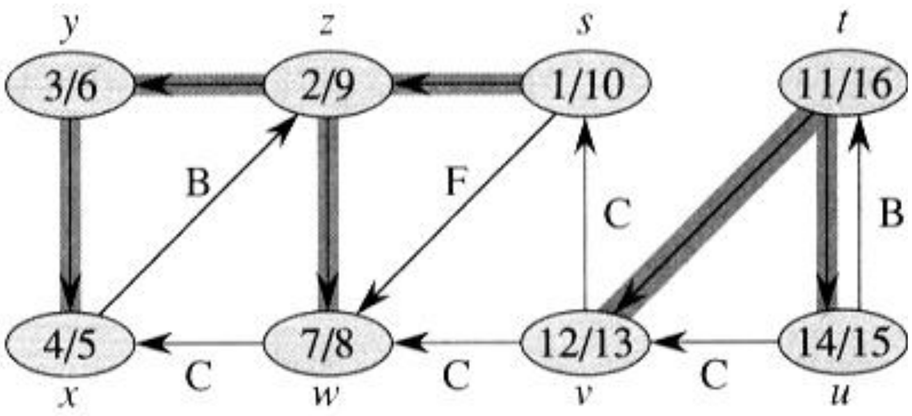
In any depth-first search of a (directed or undirected) graph $G = (V, E)$ for any two vertices u and v, exactly one of the following conditions holds:

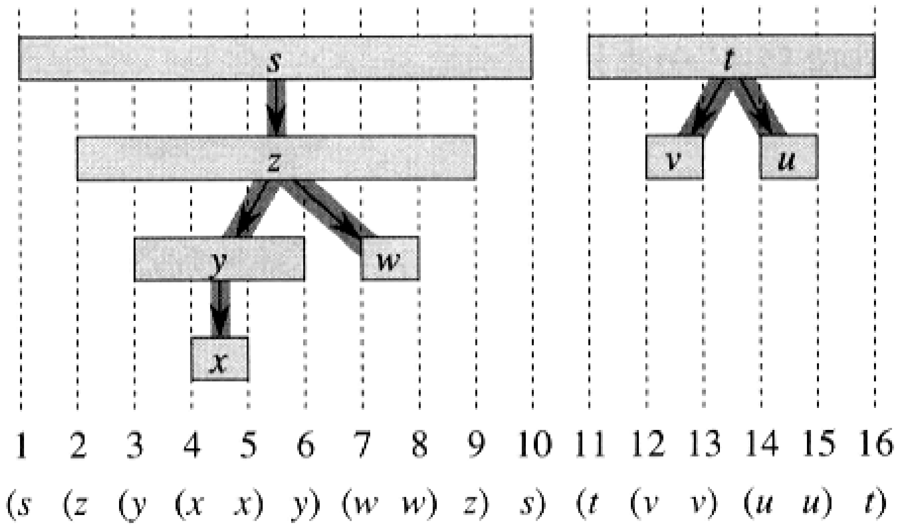
- the intervals $[d(u), f(u)]$ and $[d(v), f(v)]$ are entirely disjoint.
- the interval $[d(u), f(u)]$ is contained entirely within the interval $[d(v), f(v)]$, and u is a descendant of v in the depth-first tree, or
- the interval $[d(v), f(v)]$ is contained entirely within the interval $[d(u), f(u)]$, and v is a descendant of u in the depth-first tree.

Corollary 22.8. (Nesting of descendants' interval)

Vertex *v* is a proper descendant of a vertex *u* in the depth-first forest for a (directed or undirected) graph *G* if and only if $d(u) < d(v) < f(v) < f(u)$.

Property of DFS





Theorem 22.9 (white path theorem)

In a depth-first forest of a (directed or undirected) graph $G = (V, E)$, vertex v is a descendant of vertex u if and only if at time $d[u]$ that the search discover u, vertex can be reached from u along a path consisting entirely of white vertices.

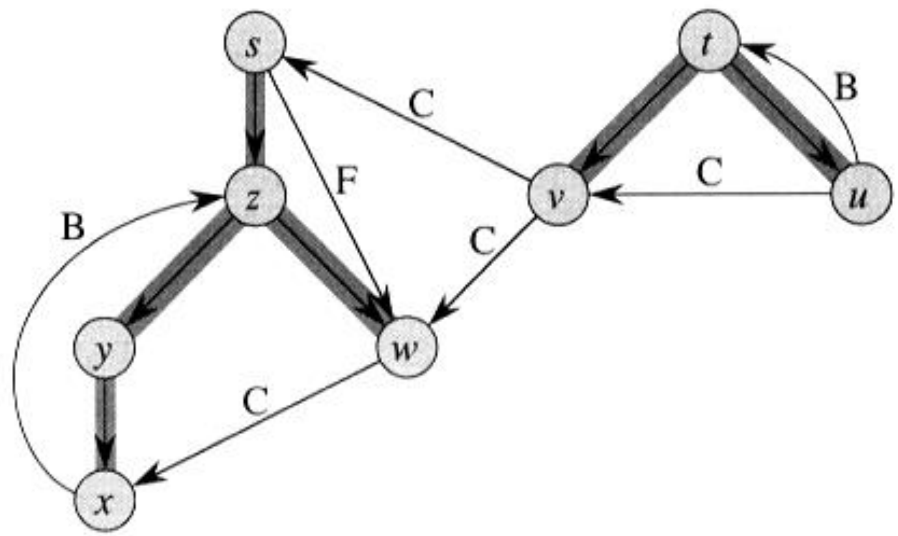
Classification of edges:

Tree edges (shaded)

Back edges

Forward edges

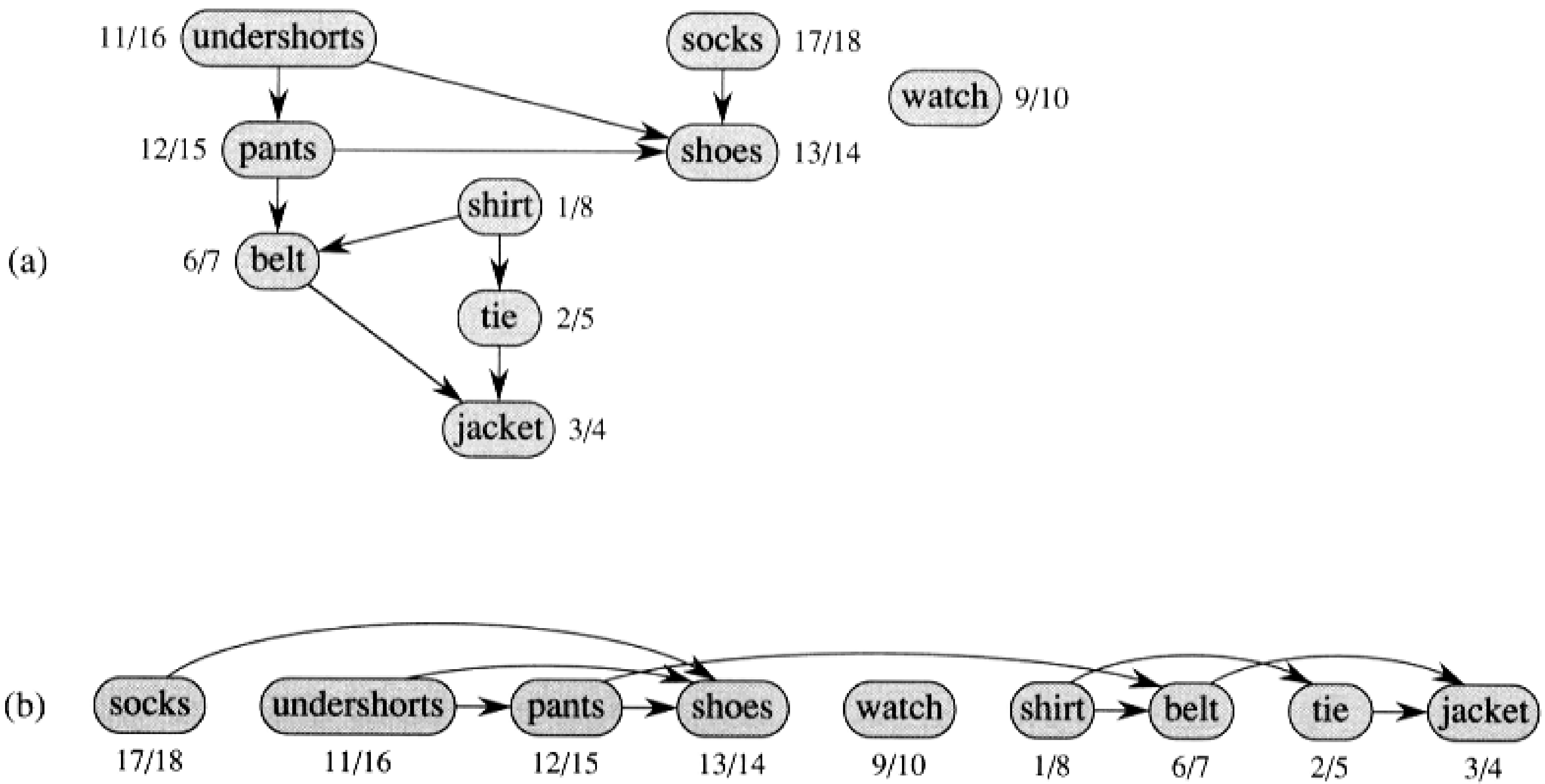
Cross edges



Theorem 22.10. In a depth-first search of an undirected graph G every edge of G is either a tree edge or a back edge.

22.4 Topological sort

A topological sort of a directed acyclic graphs $G = (V, E)$ 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.


(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

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

Lemma 22.11. A directed graph G is acyclic if and only if a depth first search of G yields no back edge.

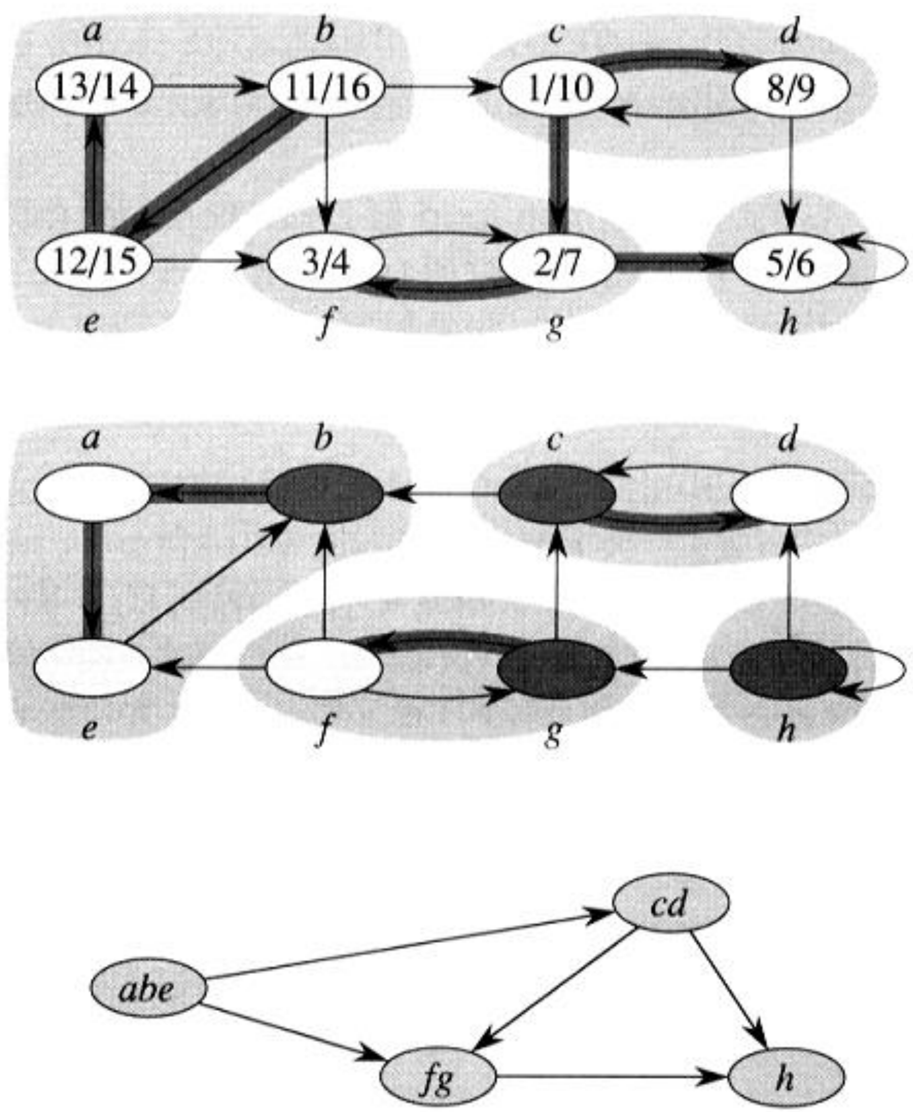
Theorem 22.12. TOPOLOGICAL_SORT(G) produces a topological sort of a directed acyclic graph G.

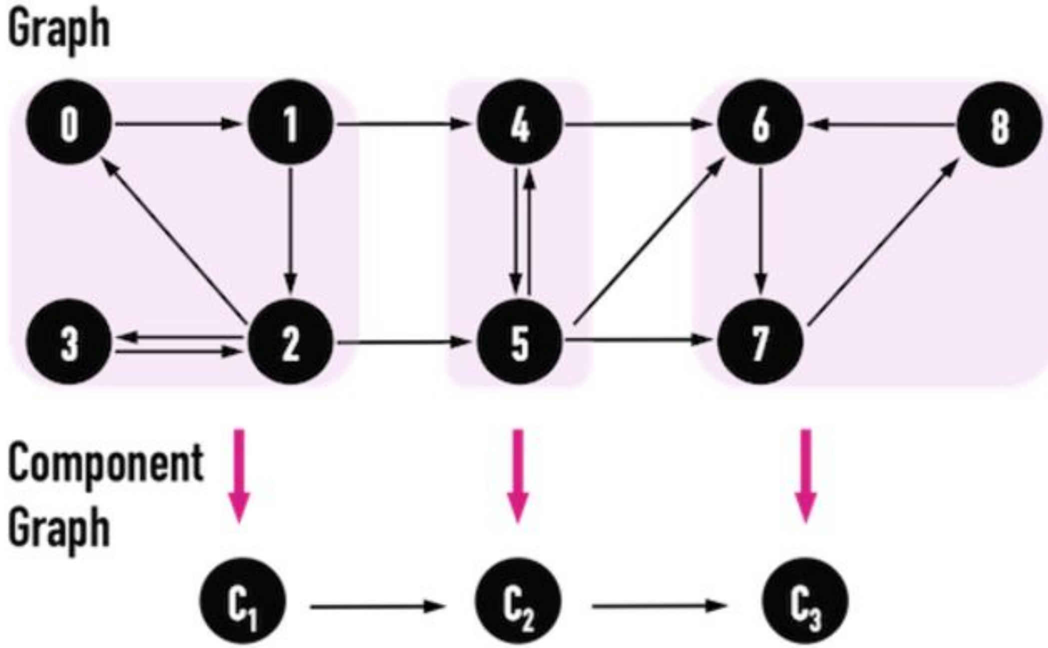
22.5 Strongly connected components

(a)

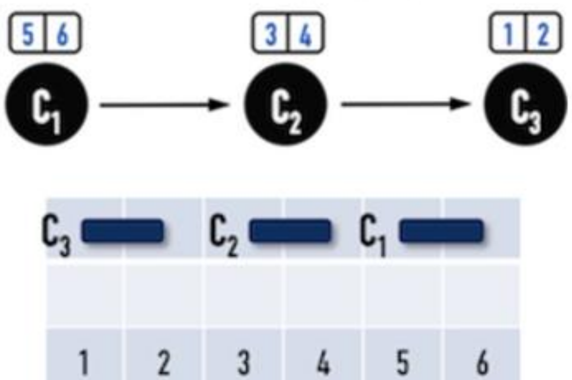
(b)

(c)

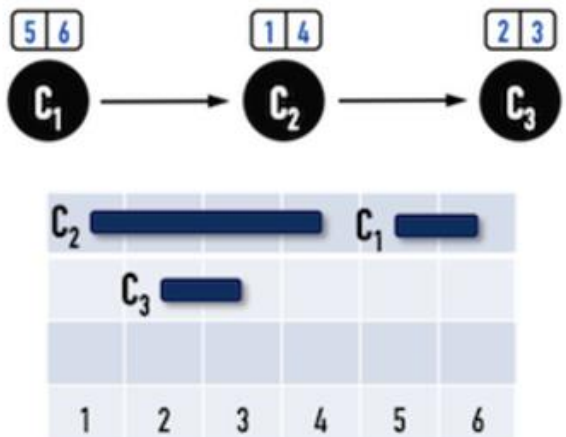




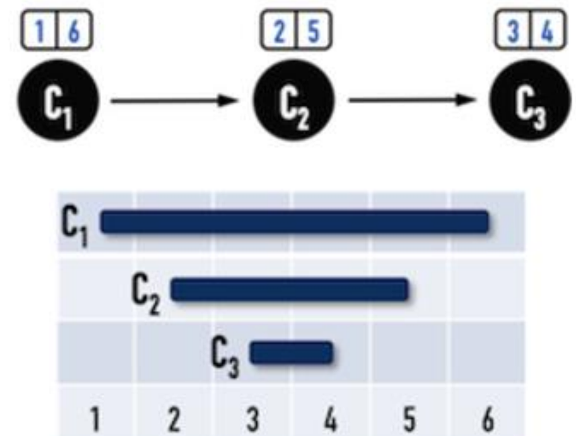
Starting Vertex: C_3, C_2, C_1



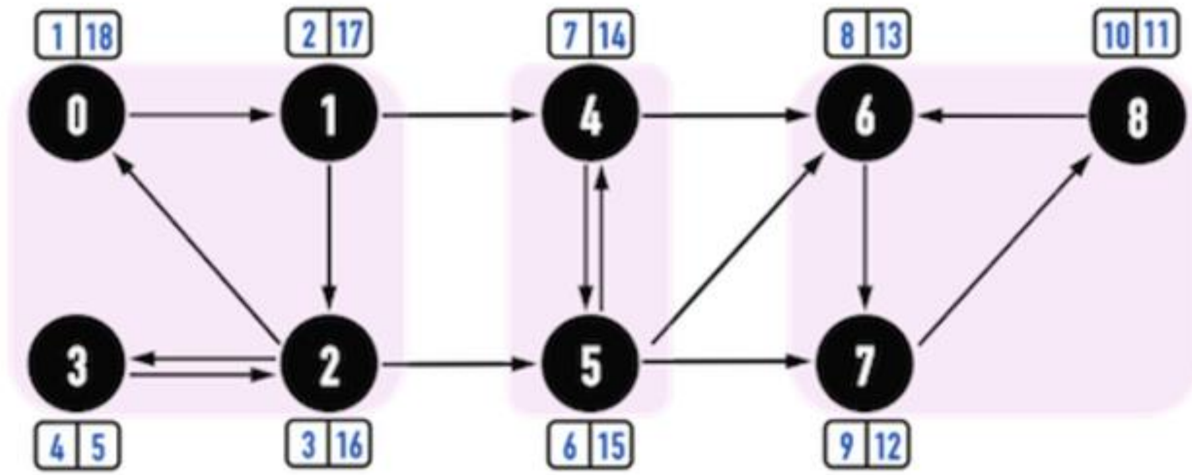
Starting Vertex: C_2, C_3, C_1



Starting Vertex: C_1, C_2, C_3

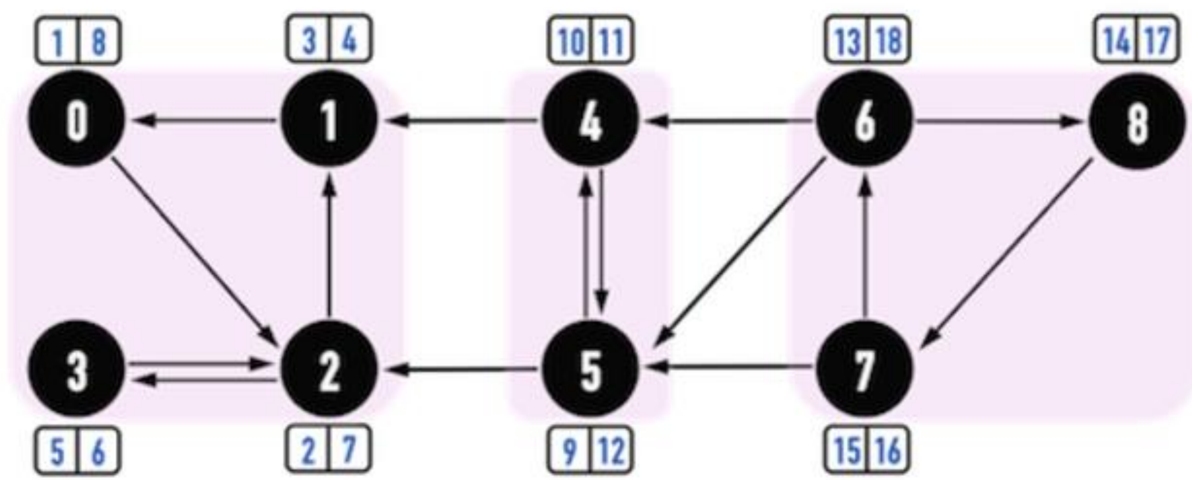


DFS(G) on vertex(0)



finish time: 0>1>2>5>4>6>7>8>3

DFS(G^T) in order of decreasing finish time



22.5 Strongly connected components

1) call $DFS(G)$ to compute finishing times $f[u]$ for each vertex u
2) compute G^T
3) call $DFS(G^T)$, but in the main loop of DFS, consider the vertices in order of decreasing $f[u]$ (as computed in line 1)
4) output the vertices of each tree in the depth-first forest of step 3 as a separate strongly connected component

Lemma 22.13. Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$, let u, $v \in C$, let u', $v' \in C'$, and suppose that there is a path $u \rightsquigarrow u'$ in G. Then there cannot also be a path $v' \rightsquigarrow v$ in G.

Lemma 22.14. Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E$, where $u \in C$ and $v \in C'$. Then $f(C) > f(C')$.

Corollary 22.15. Let C and C' be distinct strongly connected components in directed graph $G = (V, E)$. Suppose that there is an edge $(u, v) \in E^T$, where $u \in C$ and $v \in C'$. Then $f(C) < f(C')$.

Theorem 22.16.
STRONGLY-CONNECTED-COMPONENTS(G) correctly computes the strongly connected components of a directed graph G.