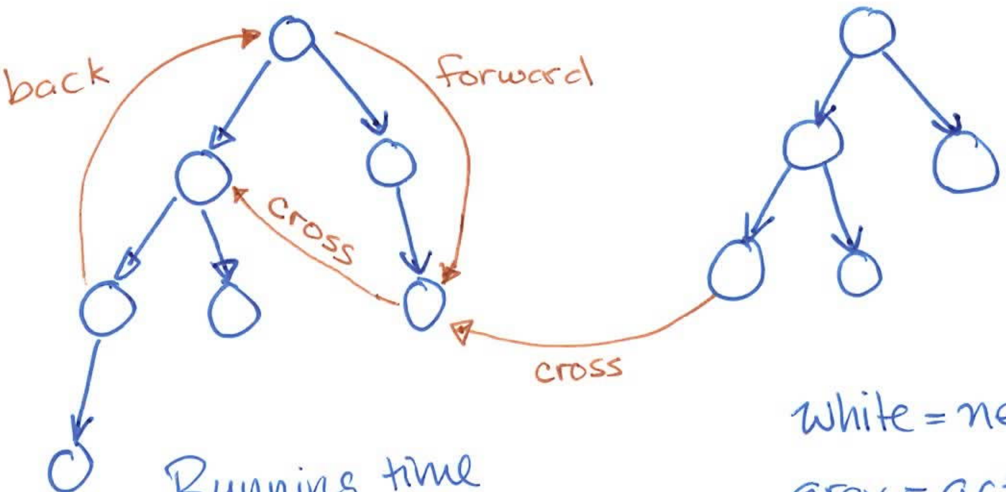
chang@umbc.edu

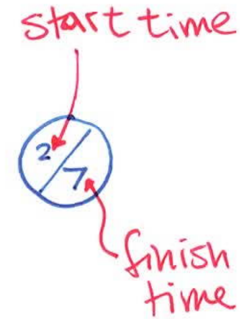
Depth-First Search

updated: 4/7/11

Depth-First Search = DFS

"Go deep"



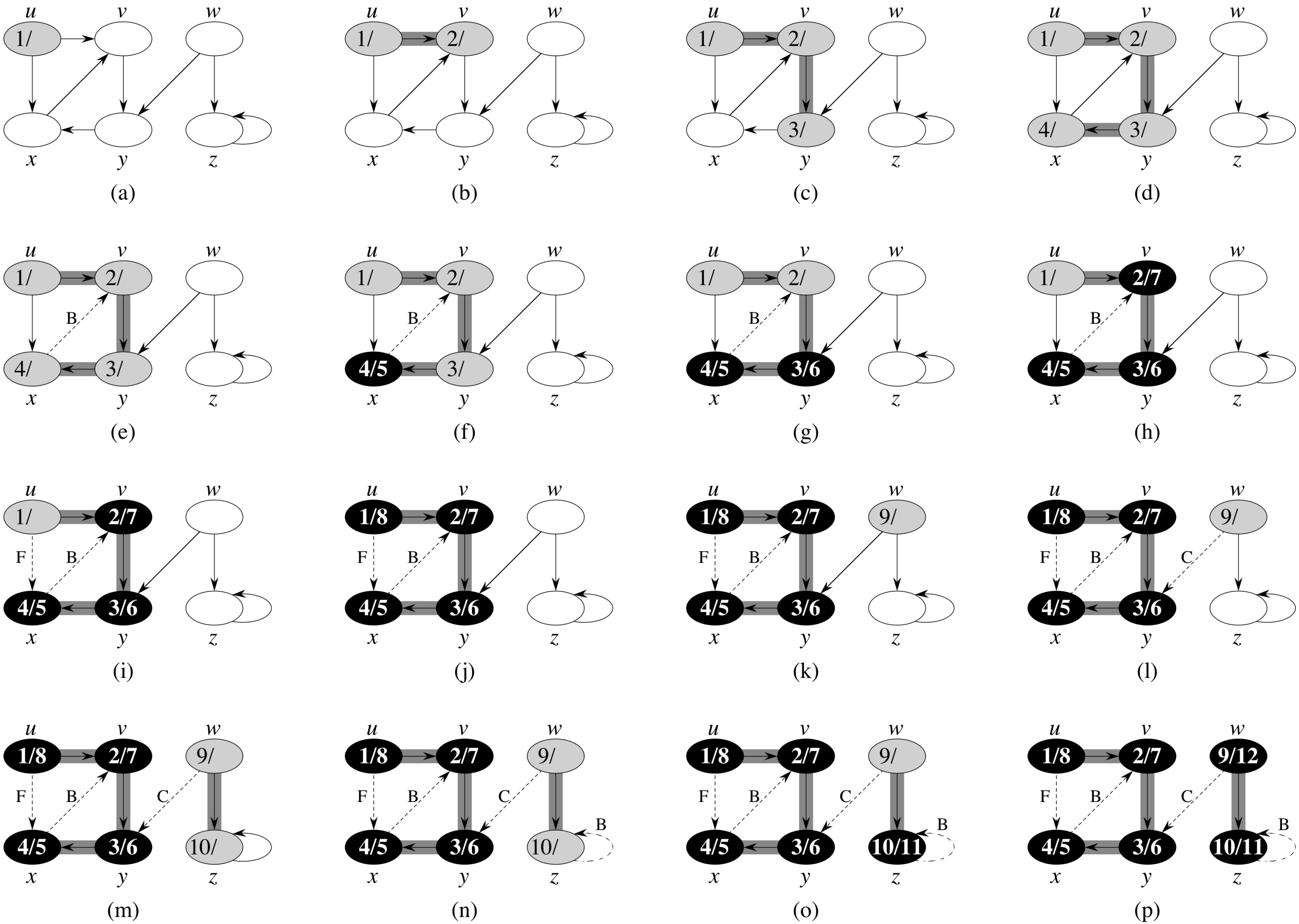


Running time
$= \Theta(V + E)$

white = new

grey = active (still investigating descendants)

black = all done



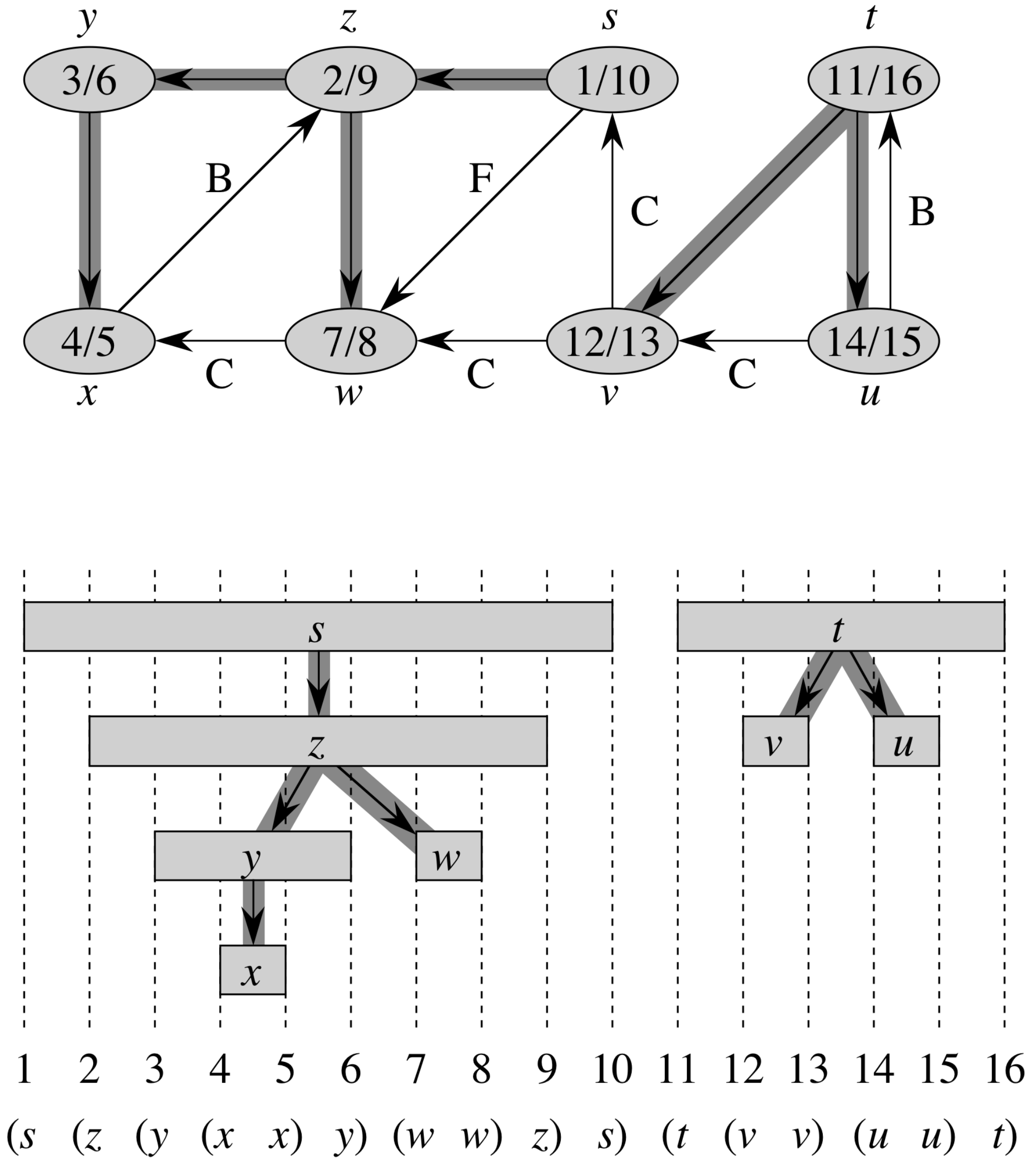
u v w
1/
x y z
(a)
u v w
1/ 2/
x y z
(b)
u v w
1/ 2/
3/
x y z
(c)
u v w
1/ 2/
4/ 3/
x y z
(d)
u v w
1/ 2/
B
4/ 3/
x y z
(e)
u v w
1/ 2/
B
4/5 3/
x y z
(f)
u v w
1/ 2/
B
4/5 3/6
x y z
(g)
u v w
1/ 2/7
B
4/5 3/6
x y z
(h)
u v w
1/ 2/7
F B
4/5 3/6
x y z
(i)
u v w
1/8 2/7
F B
4/5 3/6
x y z
(j)
u v w
1/8 2/7 9/
F B
4/5 3/6
x y z
(k)
u v w
1/8 2/7 9/
F B C
4/5 3/6
x y z
(l)
u v w
1/8 2/7 9/
F B C
4/5 3/6 10/
x y z
(m)
u v w
1/8 2/7 9/
F B C B
4/5 3/6 10/
x y z
(n)
u v w
1/8 2/7 9/
F B C B
4/5 3/6 10/11
x y z
(o)
u v w
1/8 2/7 9/12
F B C B
4/5 3/6 10/11
x y z
(p)

```
DFS(G)
for each u ∈ G.V
    u.color = WHITE
time = 0
for each u ∈ G.V
    if u.color == WHITE
        DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
time = time + 1
u.d = time
u.color = GRAY                    // discover u
for each v ∈ G.Adj[u]             // explore (u, v)
    if v.color == WHITE
        DFS-VISIT(v)
u.color = BLACK
time = time + 1
u.f = time                        // finish u
```

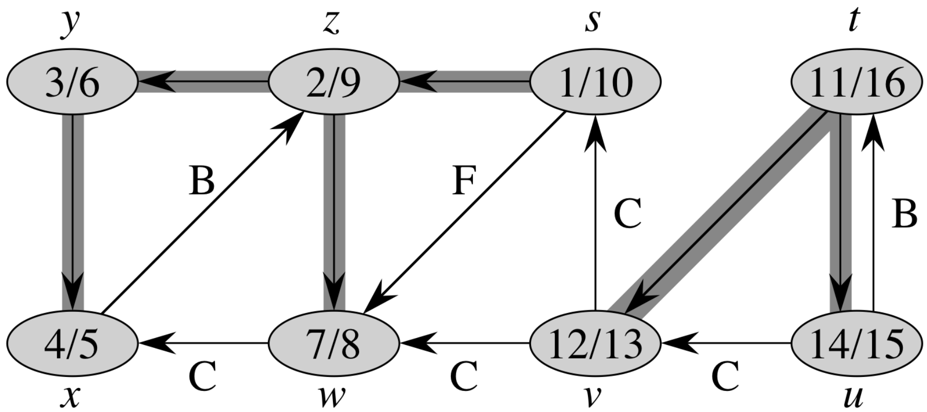
(a)

(b)

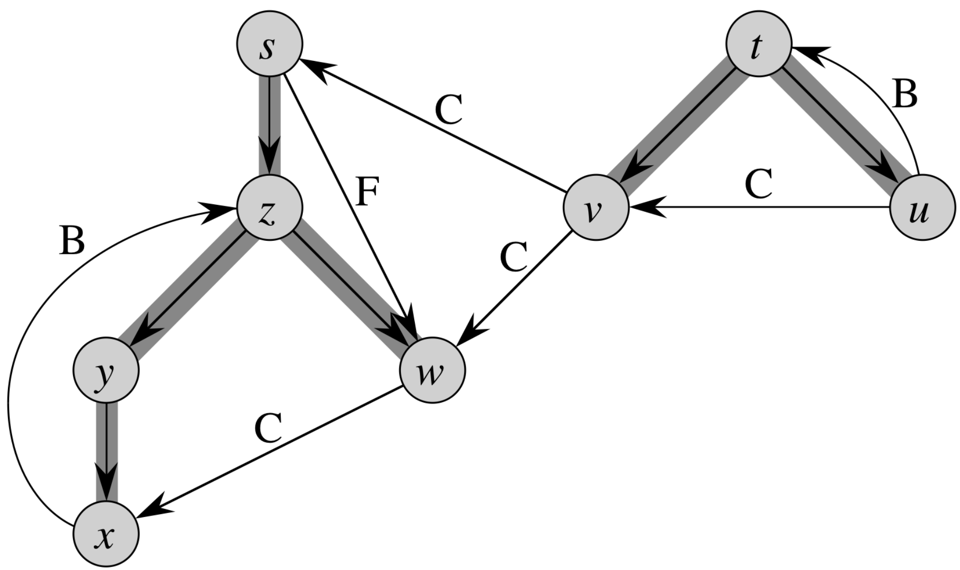


y
z
s
t
3/6
2/9
1/10
11/16
B
F
C
B
4/5
7/8
12/13
14/15
C
C
C
x
w
v
u
s
t
z
v
u
y
w
x
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
(s (z (y (x x) y) (w w) z) s) (t (v v) (u u) t)

(a)


y
z
s
t
3/6
2/9
1/10
11/16
B
F
C
B
4/5
7/8
12/13
14/15
C
C
C
x
w
v
u

(c)


s
t
C
B
F
C
z
v
u
B
C
y
w
C
x

DFS Observations:

① When vertex u is discovered, the grey vertices are exactly the ancestors of u.

② u is an ancestor of v

$$\iff d[u] < d[v] < f[v] < f[u]$$

③ Can never have

$$d[u] < d[v] < f[u] < f[v]$$

Criss-cross not allowed

④ During DFS-visit at vertex u,
when edge (u,v) is explored &
the color of v is checked

v is white $\Rightarrow$ (u,v) is a tree edge

v is grey $\Rightarrow$ (u,v) is a back edge

v is black $\Rightarrow$ (u,v) is a forward or cross edge

White Path Theorem

v is a descendant of u

$\Longleftrightarrow$

at time $d[u]$, there is a path going through white vertices from u to v.

Pf:

($\Rightarrow$) use tree edges.

White Path Theorem (cont'd)

($\Leftarrow$) Suppose at time $d[u]$, there is a white path from u to v, but v is not a descendant of u..

Let v be the first such vertex on a white path.

$$u \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet \cdots \bullet \longrightarrow w \longrightarrow v$$

all white at time $d[u]$.

Then w is a descendant of u, so

$$d[u] < d[w] < f[w] < f[u].$$

At time $d[w]$,

v was discovered already

if v is not white, then $d[u] < d[v] < d[w]$

and we have

$$d[u] < d[v] < d[w] < f[w] < f[v] < f[u]$$

and v is a descendant of u

an edge

If v is white, then edge (w,v) must be examined before $f[w]$.

If v is still white when (w,v) is examined, then (w,v) becomes a tree edge.

Otherwise, $d[w] < d[v] < f[w]$ and we have

$$d[w] < d[v] < f[v] < f[w]$$

and

$$d[u] < d[w] < d[v] < f[v] < f[w] < f[u].$$

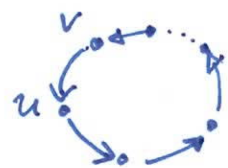
Thus, v is a descendant of u.

An application of the White Path Theorem:

Lemma G has a cycle $\iff$ DFS(G) has a back edge.

($\Leftarrow$) Easy. Suppose we have back edge (u,v). Cycle formed by tree edges from v to u and back edge from u to v.

($\Rightarrow$) Consider the cycle:



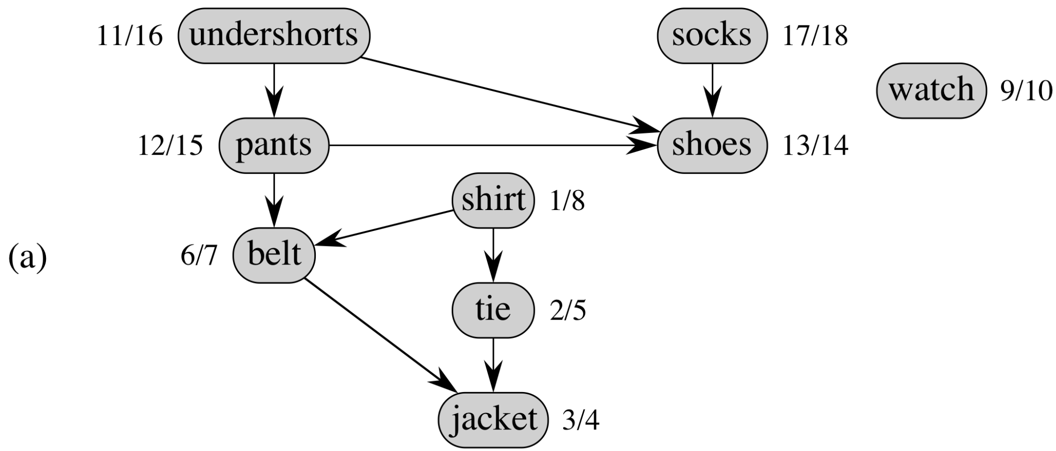
Let u have smallest $d[u]$.

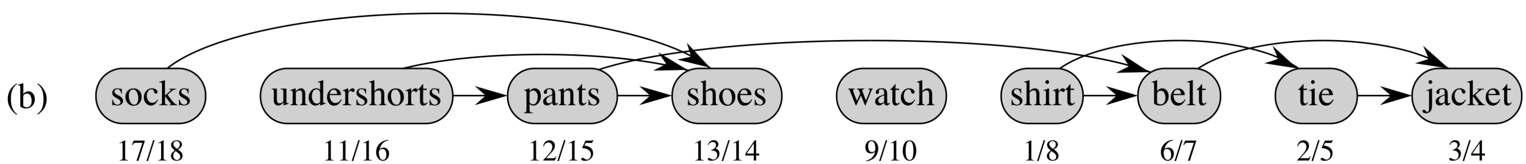
There is a white path from u to v. (Other vertices not yet discovered.) So, by White Path Theorem, v becomes a descendant of u. Thus, (v,u) is a back edge.

$\boxtimes$

Topological Sort

- directed graphs
- order vertices linearly s.t. all edges go from left to right.
- turn partial order into total order





TOPOLOGICAL-SORT(G)

call DFS(G) to compute finishing times $v.f$ for all $v \in G.V$
output vertices in order of *decreasing* finishing times

the program, not the idea

Why does TOPOLOGICAL-SORT(G) work?

Vertices are ordered from largest to smallest finish time.

Consider an edge (u,v): 

at time $d[u]$

What is the color of v when u is discovered?

All cases are OK

① v is grey: back edge, loop found, abort. OK

② v is white: v is a descendant of u (w.p.t.)

$\Rightarrow f[v] < f[u]$. OK

③ v is black: v is finished $\Rightarrow f[v] < d[u]$

$\Rightarrow f[v] < f[u]$ OK

STRONGLY CONNECTED COMPONENTS

In undirected graphs:

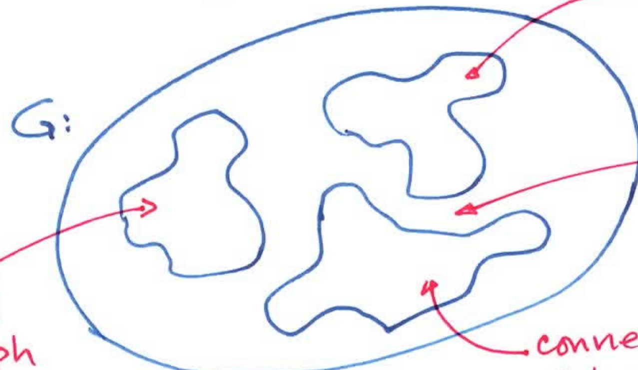
there is a path from u to v

connected : $\forall u, v \in V,\ u \rightsquigarrow v$

is also a path from v to u

connected components:

Can use BFS to find connected components



What does "connected" mean in a directed graph?

Directed Graphs:

semi-connected: $\forall u,v \in V$, either $u \rightsquigarrow v$ or $v \rightsquigarrow u$

strongly-connected: $\forall u,v \in V$, $u \rightsquigarrow v$ and $v \rightsquigarrow u$

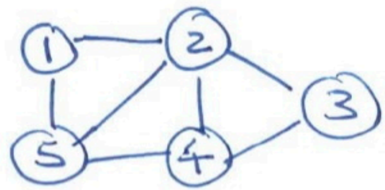
$\equiv \forall u,v \in V$, $u \rightsquigarrow v$, Why?

An induced subgraph S of G is called a <u>strongly connected component</u> of G, if S is strongly connected and $\forall$ subgraphs $S' \supsetneq S$, S' is not strongly connected.

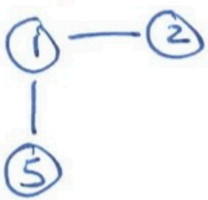
Problem: Given G, find its strongly connected components.

"Subgraphs" v.s. "Induced Subgraphs"

$G = (V, E)$



Subgraph:

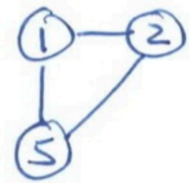


$S_1 = (V_1, E_1)$

$V_1 \subseteq V$

$E_1 \subseteq E$

Induced subgraph:



$S_2 = (V_2, E_2)$

$V_2 \subseteq V$

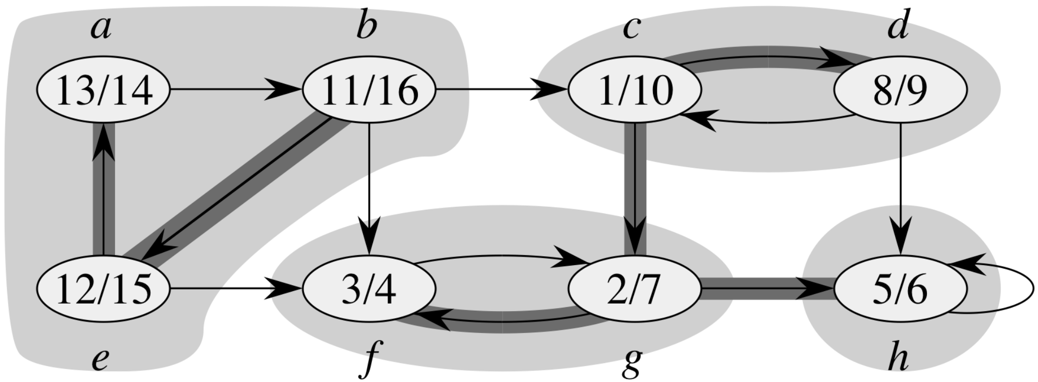
$E_2 = \{(u,v) \mid u \in V_2 \ \& \ v \in V_2\}$

must include <u>all</u> edges that have both endpoints in V_2

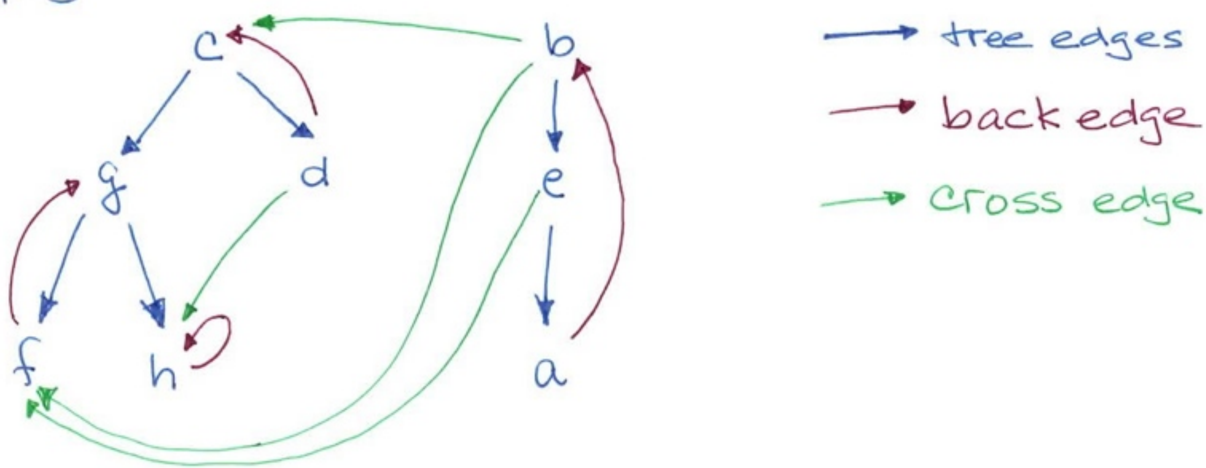
SCC(G)

call DFS(G) to compute finishing times $u.f$ for all u
compute G^{T}
call DFS(G^{T}), but in the main loop, consider vertices in order of decreasing $u.f$
(as computed in first DFS)
output the vertices in each tree of the depth-first forest formed in second DFS
as a separate SCC

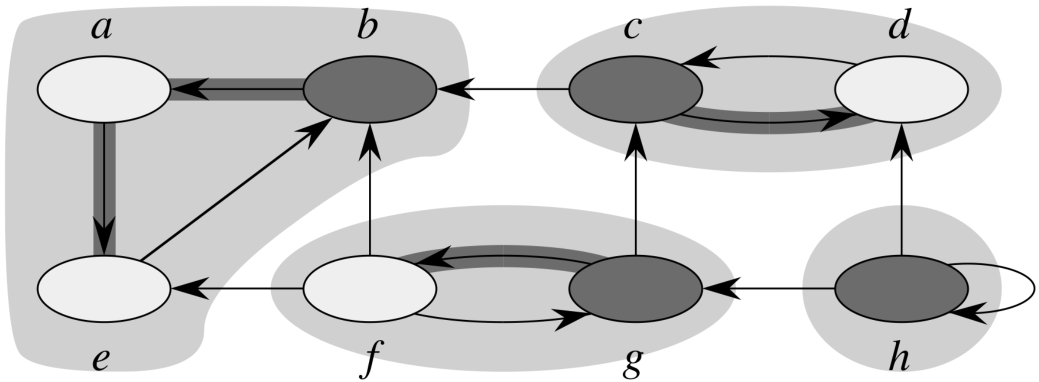
(a)


a
b
c
d
13/14
11/16
1/10
8/9
12/15
3/4
2/7
5/6
e
f
g
h

1st DFS


c
b
g
d
e
f
h
a
tree edges
back edge
cross edge

(b)



$\boxed{\text{2nd DFS}}$

f[ ] values from first DFS

b	e	a	c	d	g	h	f
16	15	14	10	9	7	6	4

DFS trees

$b \rightarrow a \rightarrow e$

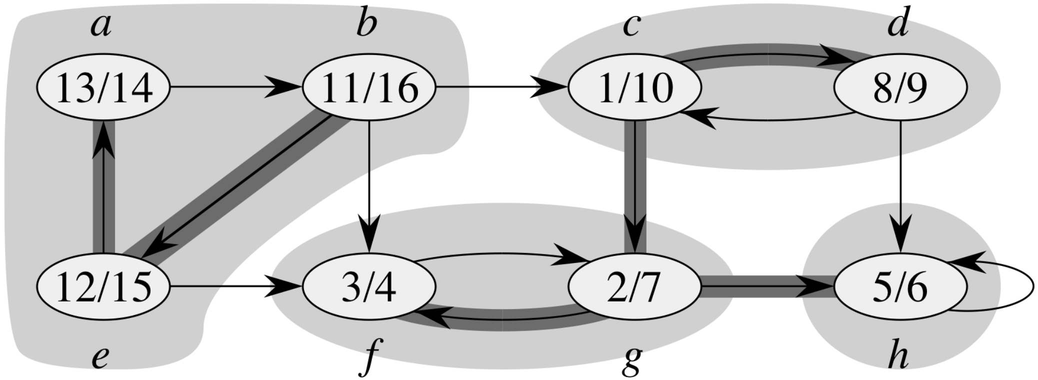
$c \rightarrow d$

$g \rightarrow f$

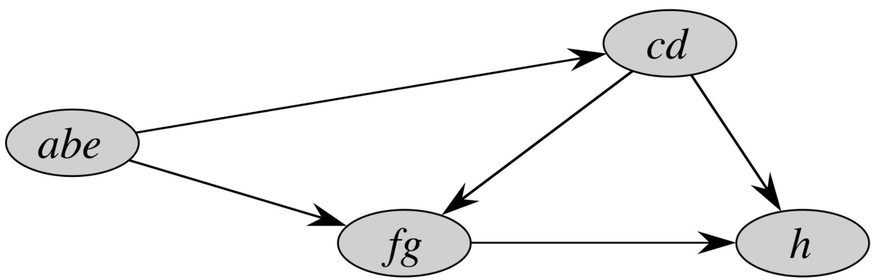
h

back edges & cross edges not shown

(a)



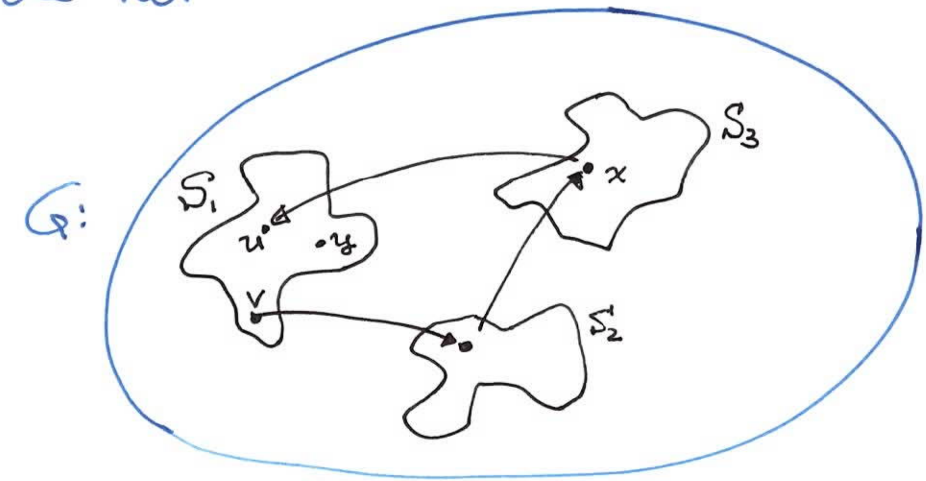
(c)



The component graph must be acyclic.

Why?

Suppose not



$\forall y \in S_1$, $\quad x \rightsquigarrow u$ and $u \rightsquigarrow y \Rightarrow x \rightsquigarrow y$

$\quad y \rightsquigarrow v$ and $v \rightsquigarrow x \Rightarrow y \rightsquigarrow x$

Thus, x should be in S_1 $\Rightarrow\!\Leftarrow$

Why does SCC(G) work?

Lemma 1: If v & w are vertices in the same s.c.c., then in any DFS of G, v & w are in the same DFS tree.

Pf: Let S be the s.c.c. that contains v & w.
Fix a DFS of G.
Let $x \in S$ be the first vertex of S to be discovered.
$x \rightsquigarrow v$ & $x \rightsquigarrow w$ ← because they are in same s.c.c.
all vertices on these paths are white at time $d[x]$.

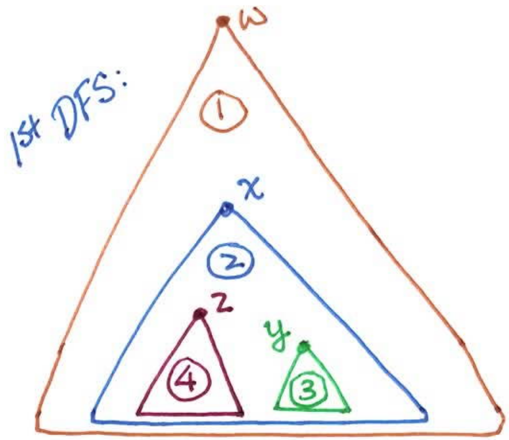
By w.p.t., v becomes a descendant of x in DFS tree.
Similarly, w " " " " .
Thus, v & w are in the same DFS tree.

Corollary: If v & w are in the same s.c.c., then in *any* DFS of G^T, v & w are in the same DFS tree.

Pf: G & G^T have the same s.c.c. $\boxtimes$

Intuitively, ...

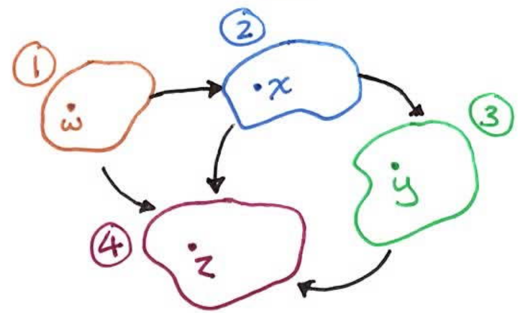
Let x be the first node of s.c.c. 2 to be discovered.



x has largest f[ ] of vertices in component 2.

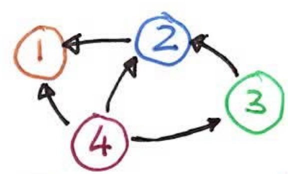
All vertices in 2 become descendants of x

component graph:

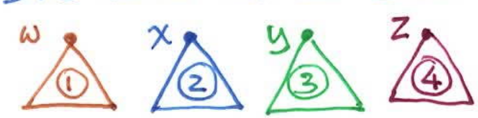


largest f[ ] first is the topological ordering of the components

component graph of G^T



DFS trees of 2nd DFS:



Lemma 2: SCC(G) works.

Pf: By Corollary, each s.c.c. is contained in one of the DFS trees of G^T.

We need to show that each DFS tree contains only one s.c.c.

Let x be the root of a DFS tree in the second DFS of G^T.
Let v be any descendant of x.

$x \overset{T}{\rightsquigarrow} v$ via tree edges, so $v \rightsquigarrow x$ in original G

Need 2 Claims: Claim 1: $f[v] < f[x]$

Claim 2: $d[x] < d[v]$

f[] & d[] are always from 1st DFS

Using Claim 1 & Claim 2, we have:

$$\underbrace{d[x] < d[v]}_{\text{Claim 2}} < \underbrace{f[v] < f[x]}_{\text{Claim 1}}$$

So, v is a descendant of x in 1st DFS.

Thus, $x \rightsquigarrow v$.

We already know $v \rightsquigarrow x$.

Therefore, v & x are in the same s.c.c.

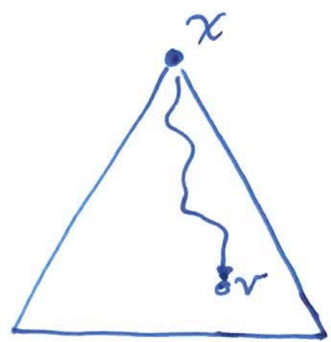
$\therefore$ Every descendant of x in 2nd DFS is in the same s.c.c. as x.

Claim 1: $f[v] < f[x]$.

$f[\,]$ values from 1^{st} DFS, not 2^{nd} DFS!!!

Pf: When x was chosen to be a root of a DFS tree in the second DFS, v was not yet discovered in the second DFS. (Otherwise, v would not become a descendant of x.)

If v had a larger finish time than x, then v would be chosen to become root.



$\therefore f[x] > f[v]$

$\boxtimes$

END OF CLAIM 1

Claim 2: $d[x] < d[v]$

cannot be equal.

Pf: (By contradiction) Suppose not. Then $d[v] < d[x]$.
v is a descendant of x in 2^{nd} DFS so

$$\underset{x}{\bullet} \to \bullet \to \bullet \to \bullet \cdots \underset{v'}{\bullet} \to \cdots \to \underset{v}{\bullet}$$

using tree edges in G^T

Let v' be the first vertex on this path with $d[v'] < d[x]$.

Maybe $v' = v$.

Since we used tree edges, v' is also a descendant of x.

$$x \to \underbrace{y_1 \to y_2 \to y_3}_{\text{just gave names to these vertices}} \to \cdots \to y_r \to v'$$

Then, $d[x] < d[y_i]$ for all y_i. Otherwise, v' is not first.

So, for all y_i, $d[v'] < d[x] < d[y_i]$.

At time $d[v']$ in 1^{st} DFS, when v' was discovered, none of x or $y_1, y_2, \ldots, y_r$ had been discovered.

Look at this path in original G:

$$v' \to y_r \to y_{r-1} \to \cdots \to y_1 \to x$$

edges reversed from before

This is a white path, so by the w.p.t. x becomes a descendant of v' in the 1^{st} DFS. Thus,

$$d[v'] < d[x] < f[x] < f[v'].$$

But, v' is a descendant of x in the 2^{nd} DFS, so, by Lemma 1, $f[v'] < f[x]$. $\Rightarrow\Leftarrow$

$\therefore$ $d[x] < d[v]$