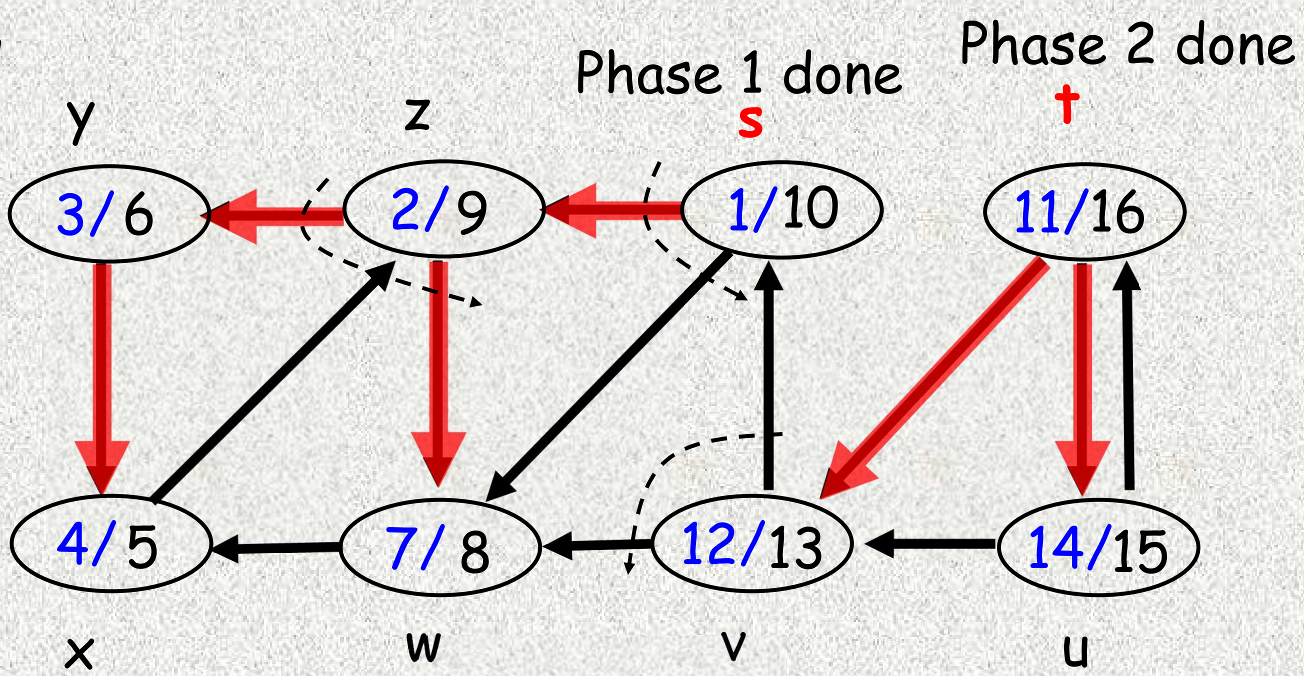
Procedure	Binary heap (worst-case)	Fibonacci heap (amortized)	array
MAKE-HEAP (empty)	$\Theta(1)$	$\Theta(1)$	O(1)
INSERT	$\Theta(\lg n)$	$\Theta(1)$	O(1)
MINIMUM	$\Theta(1)$	$\Theta(1)$	O(n)
EXTRACT-MIN	$\Theta(\lg n)$	$O(\lg n)$	O(n)
UNION	$\Theta(n)$	$\Theta(1)$	O(n)
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(1)$	O(1)
DELETE	$\Theta(\lg n)$	$O(\lg n)$	O(1)
Build	O(n)	O(n)	O(n)

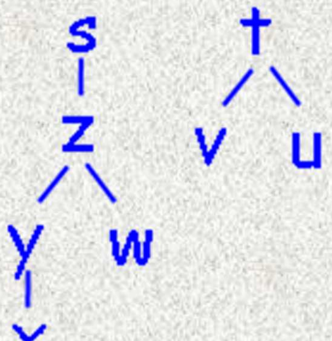
An array

(a, 4)	(b, 7)	(c, 9)	(d, ~~5~~ 3)	(e, 8)	(f, 2)	(g, 3)	(h, 6)

Extract-Min
Decrease-Key (d, 5) -> (d, 3)
Delete c

22-9 Fig





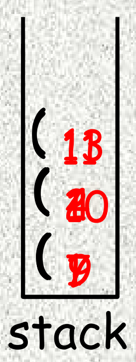
depth-first forest

Parenthesis structure: (well-formed, nested)

(( ) ( )) ( ) ((( ) ( ))) yes

(two pairs: include or disjoint)

(( ) ( ))) (((( ) ( ))) no

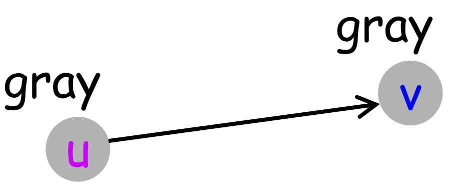


yes

(( ) ( )) ( ) ((( ) ( )))

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Case 2:



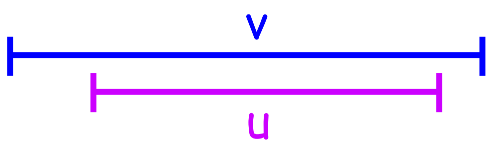
* intervals are not disjoint

⇨ descendant – ancestor relation

(nested intervals)

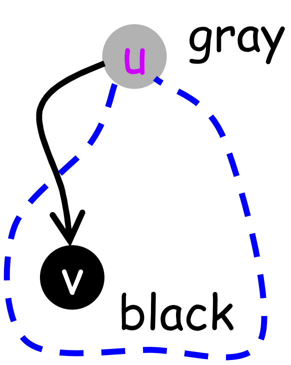
* $f(u) < f(v)$ (* u 先 結 束 *)

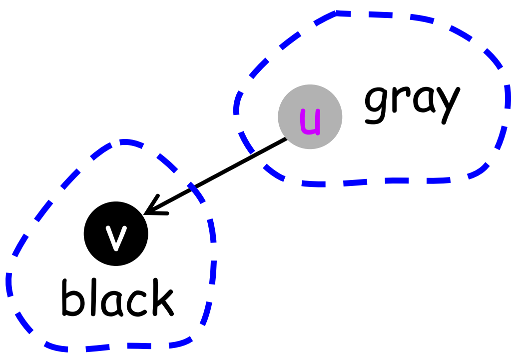
⇨



⇨ back edge !

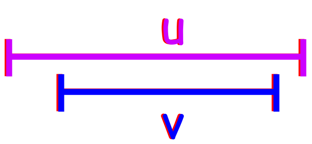
Case 3:





$f(v) < f(u)$

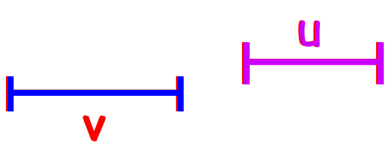
* $d(u) < d(v)$



⇨ nested

⇨ forward edge!

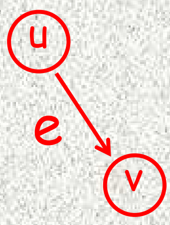
* $d(u) > d(v)$

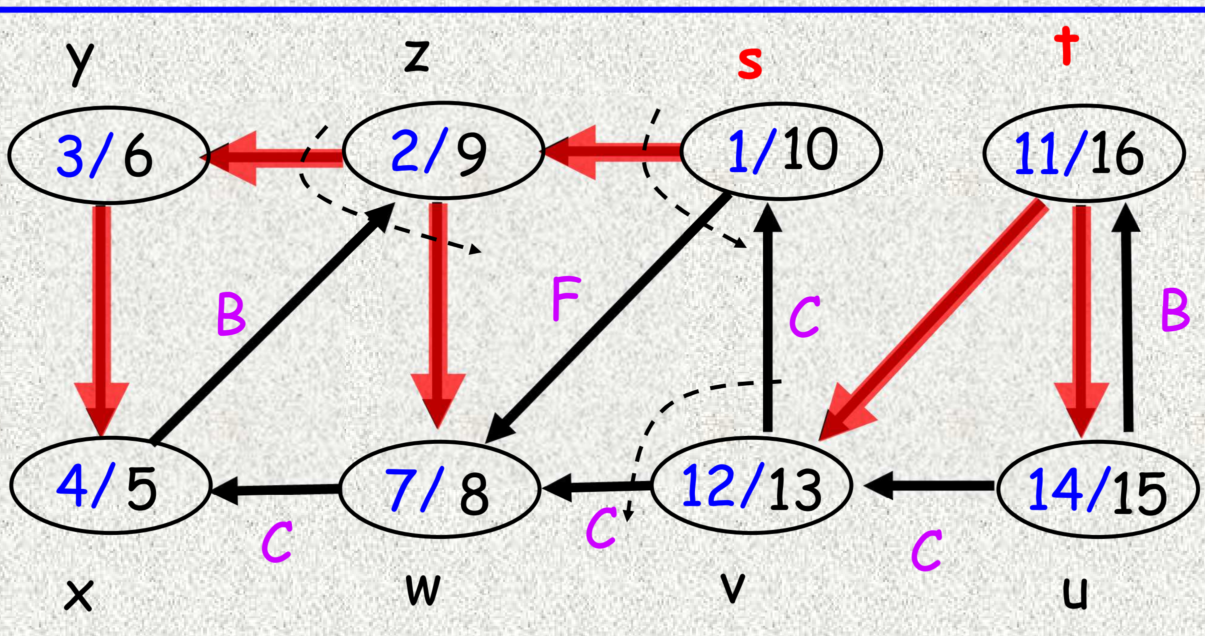


⇨ disjoint

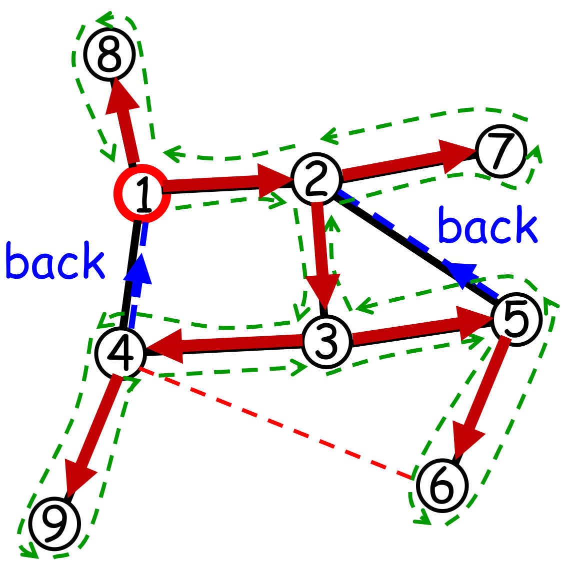
⇨ cross edge!

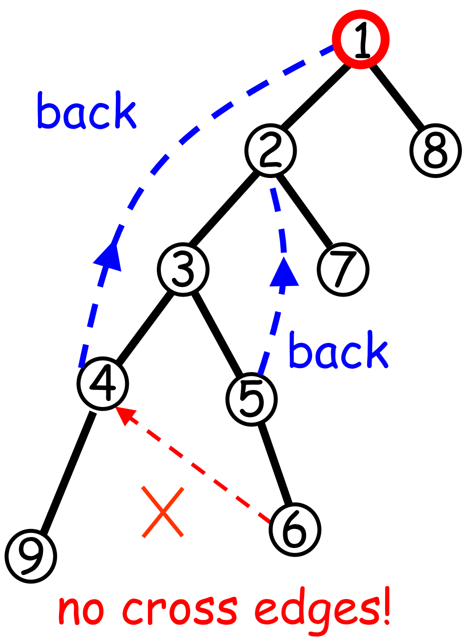
1. v is white	→	tree	
2. v is gray	→	back	
3. v is black	→	forward	if d[u] < d[v]
		cross	if d[u] > d[v]



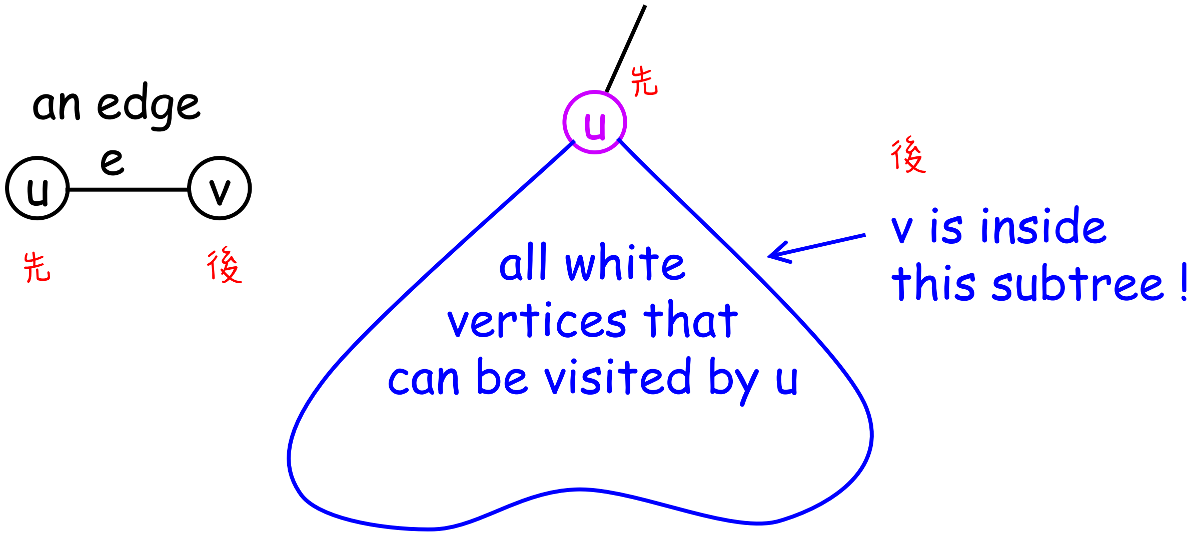


Theorem 22.10





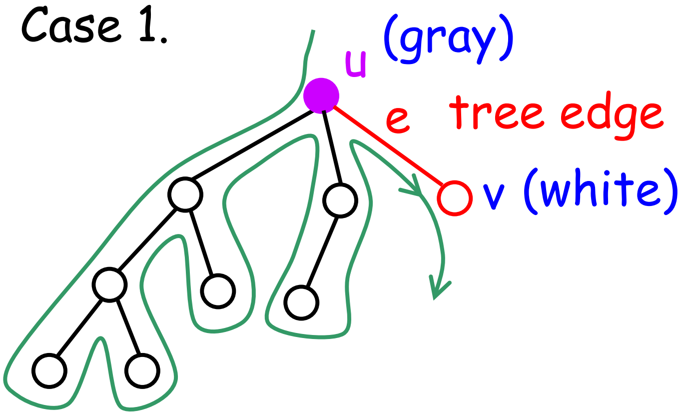
Theorem 22.10



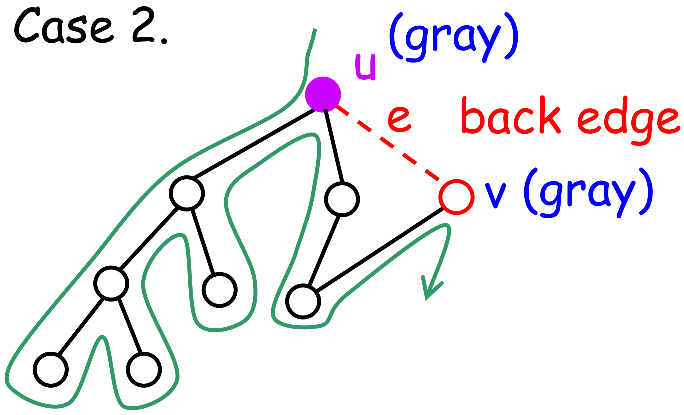
⇨ v is a descendant of u

* 在 u finish 前 , v 必 會 被 visit !!

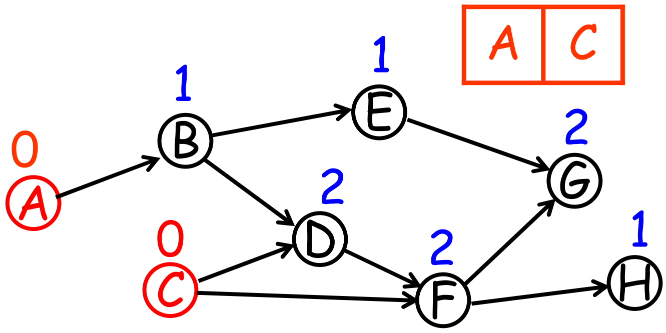
Case 1.



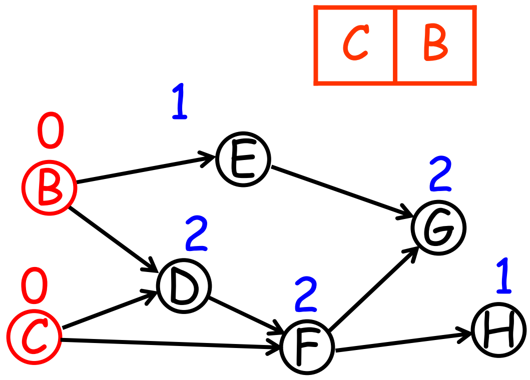
Case 2.



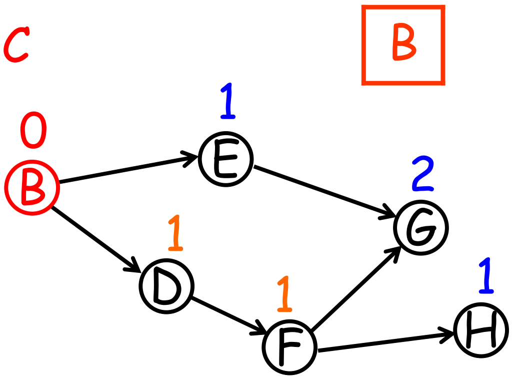
22-12a



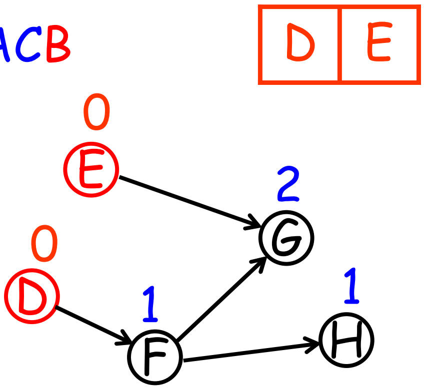
(1) A



(2) AC

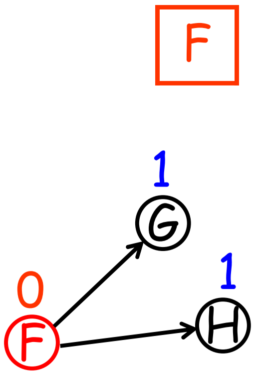


(3) ACB

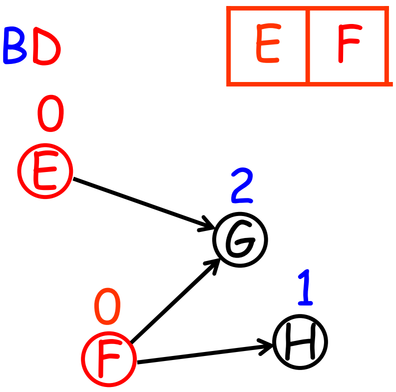


(5) ACBDE

22-12b



(4) ACBD

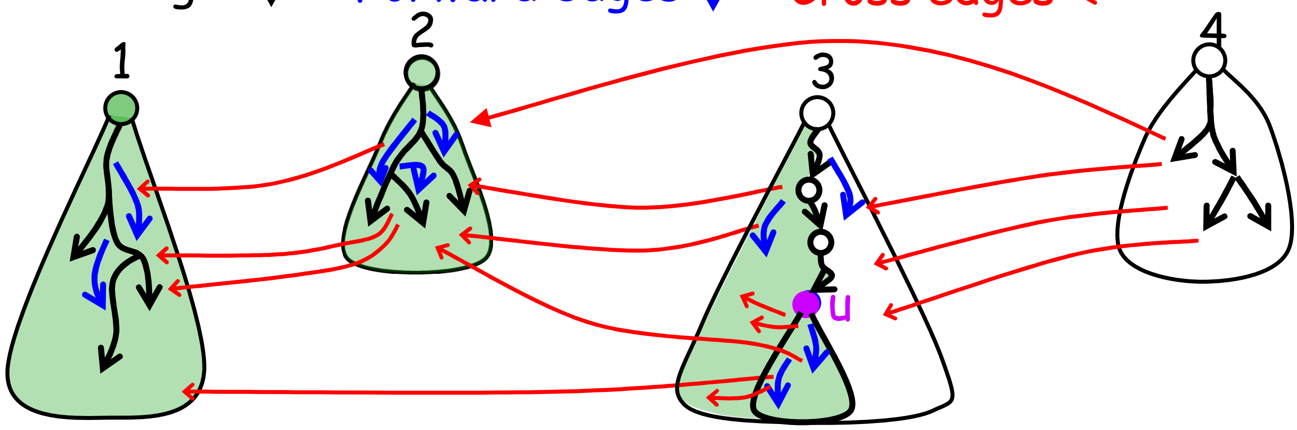


.
.
.

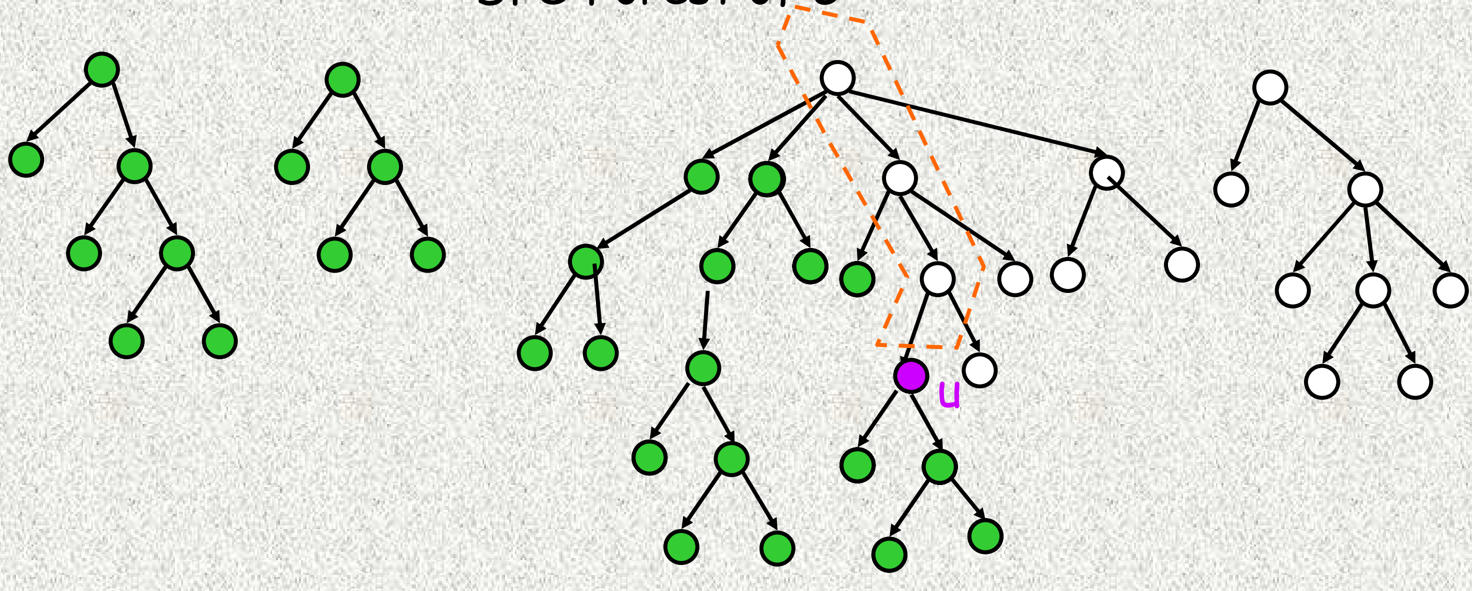
(8) ACBDEFGH

Topological sort - Correctness

All edges are:
tree edges ↓ Forward edges ↓ Cross edges ←



DFS Forest of G



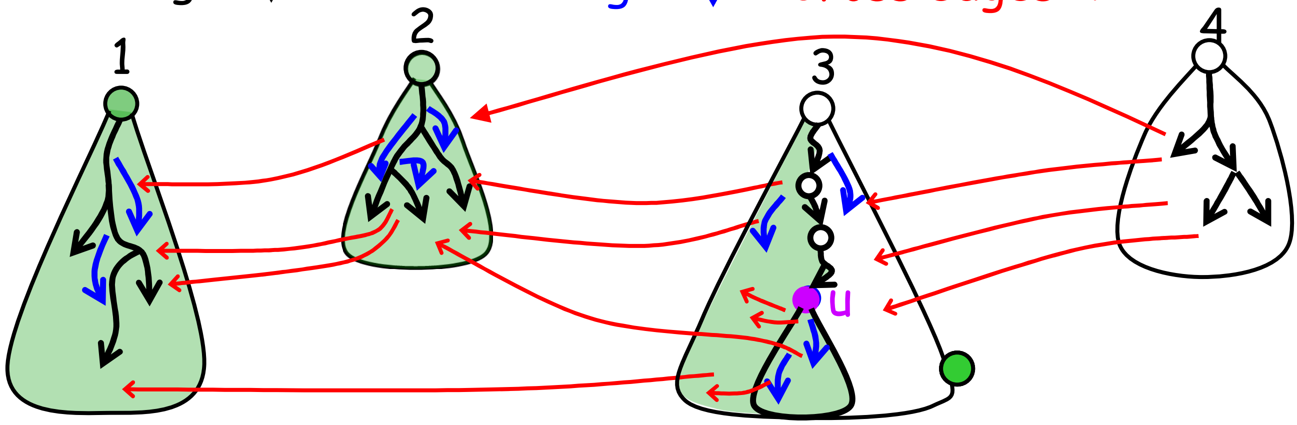
■ : f(■) < f(u) ⇨ may be arrived from u
(only downward/to-the-left edges)

□ : f(□) > f(u) ⇨ can not be arrived from u
(no upward and to-the-right edges)

All edges are: **Topological sort - Correctness**

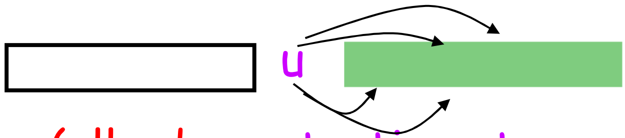
22-12c

tree edges ↓ Forward edges ↓ Cross edges ←



■ : f(■) < f(u) ⇨ may be arrived from u
(only downward/to-the-left edges)

□ : f(□) > f(u) ⇨ can not be arrived from u
(no upward and to-the-right edges)

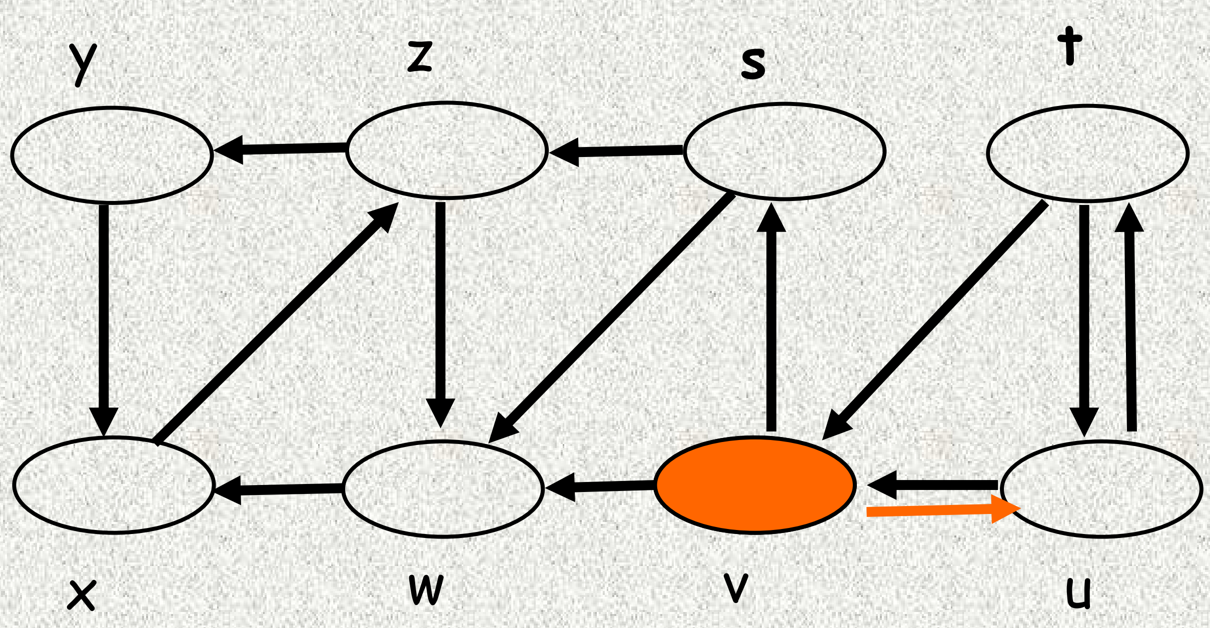
⇨ 

(in order of decreasing f(u))

(all edges starting at u are from left to right)
(all edges are from left to right)

22-9 Fig

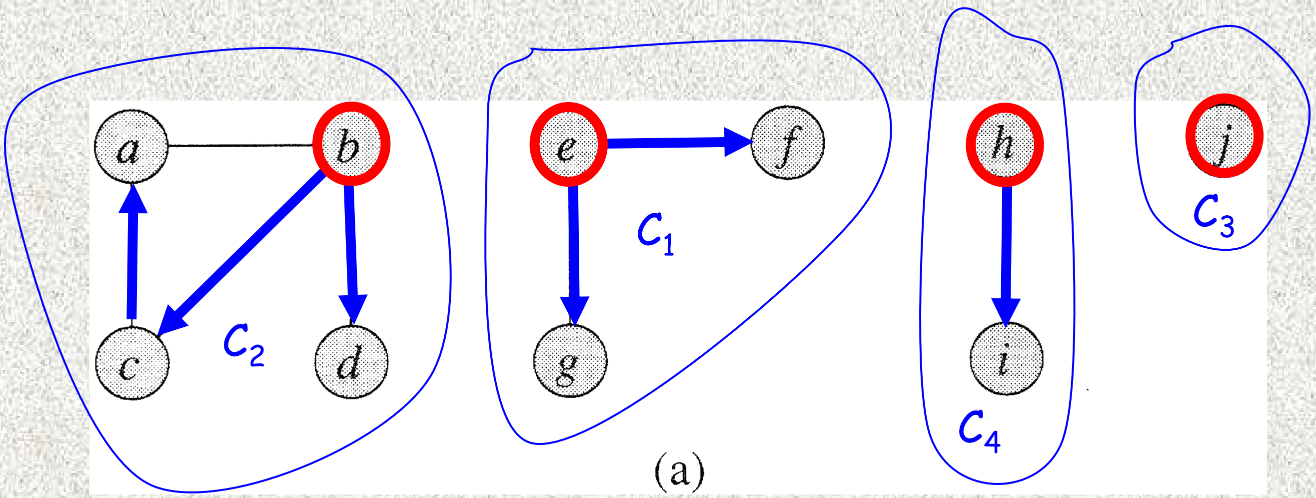
Strongly Connected components of a directed graph



22-9x

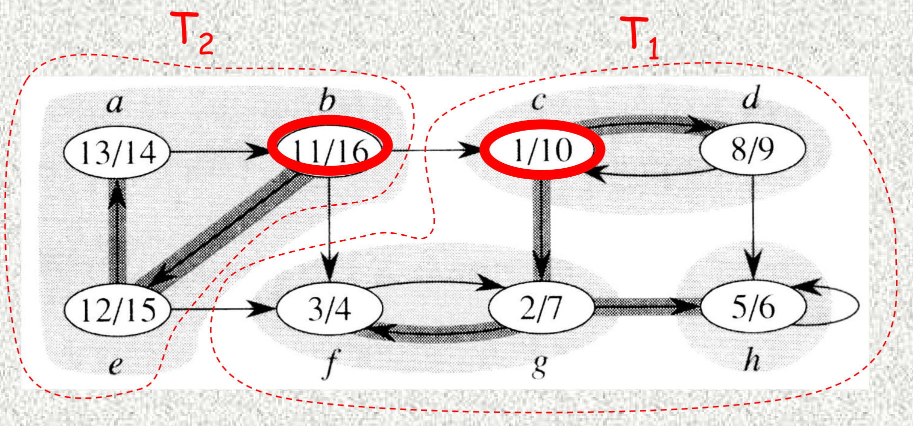
Connected components of an undirected graph

- A simple DFS is sufficient

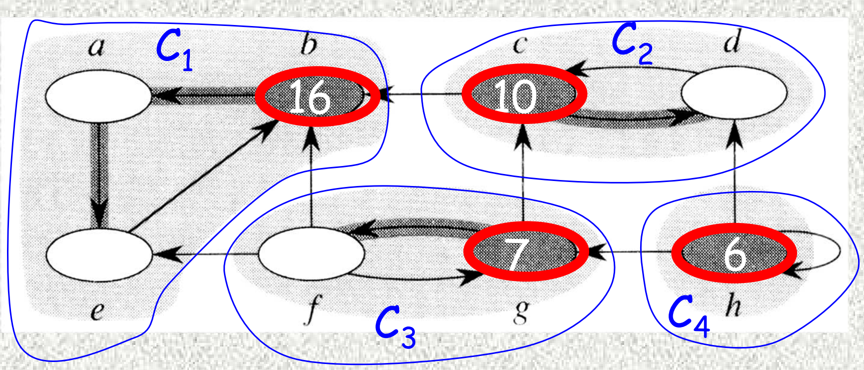


(See 21-2 Fig., application of disjoint set)

G



G^T



Phase 2 (on G^T)

sort by f[u] decreasingly first

```
DFS(G)
for each vertex u ∈ V[G]
    do color[u] ← WHITE
       π[u] ← NIL
time ← 0
for each vertex u ∈ V[G]
    do if color[u] = WHITE
          then DFS-VISIT(u)

DFS-VISIT(u)
color[u] ← GRAY              ▷ White vertex u has just been
time ← time + 1                                   discovered.
d[u] ← time
for each v ∈ Adj[u]          ▷ Explore edge (u, v).
    do if color[v] = WHITE
          then π[v] ← u
               DFS-VISIT(v)
color[u] ← BLACK             ▷ Blacken u; it is finished.
f[u] ← time ← time + 1
```