Algorithms

Lesson #5b:

DFS

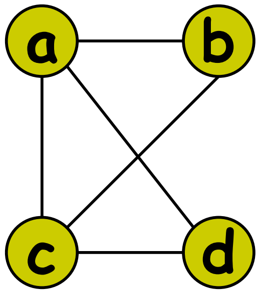
Topological Sort

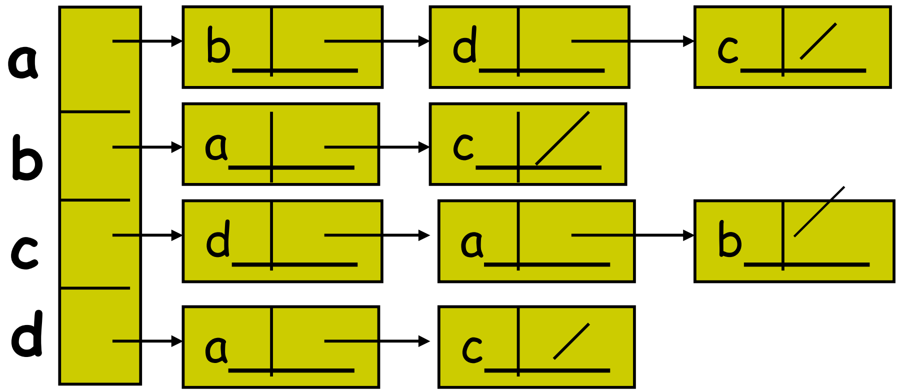
Strongly Connected Components

(Based on slides by Prof. Dana Shapira)

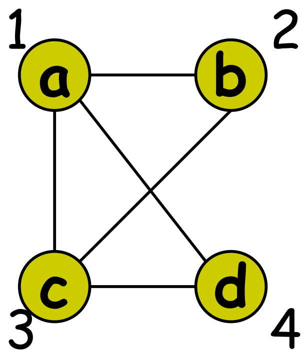
Graph Representations

- Adjacency Lists.





- Adjacency Matrix.



	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

Depth-first Search

- ***Input:*** $G = (V, E)$, directed or undirected.
- ***Output:***
  - for all $v \in V$.
    - $d[v]$ = ***discovery time*** (v turns from white to gray)
    - $f\,[v]$ = ***finishing time*** (v turns from gray to black)
  - $\pi[v]$: predecessor of $v = u$, such that v was discovered during the scan of u's adjacency list.
- ***Forest of depth-first trees:***

  $G_\pi = (V, E_\pi)$ $E_\pi = \{(\pi[v], v),\ v \in V \text{ and } \pi[v] \neq \textbf{null}\}$

- Colors the vertices to keep track of progress.
  - *White* – Undiscovered.
  - *Gray* – Discovered but not finished.
  - ***Black*** – Finished.

DFS(G)

Main Loop

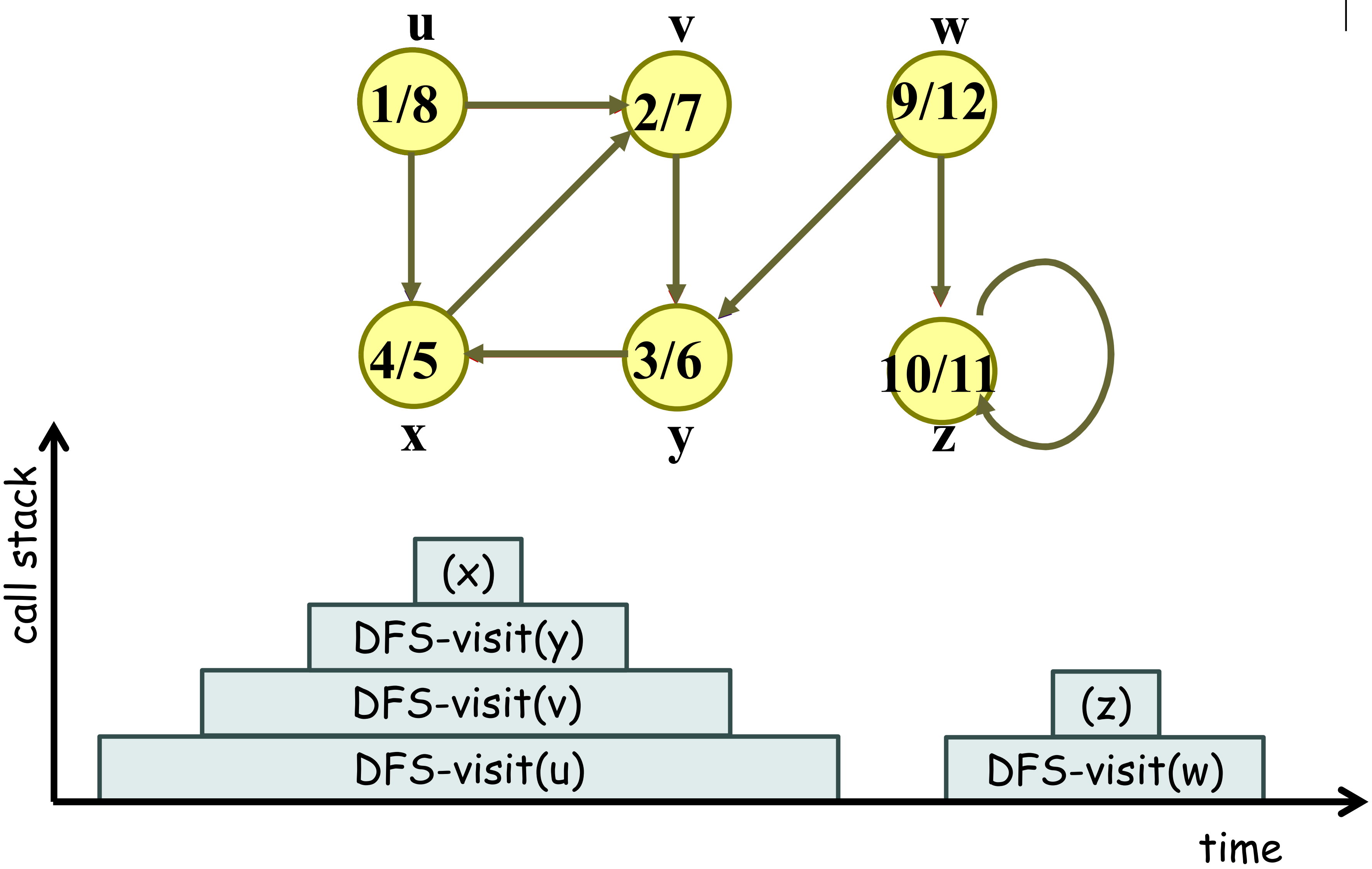
1. **for** each vertex $u \in V[G]$
2. **do** $color[u] \leftarrow$ white
3. $\pi[u] \leftarrow$ NULL
4. $time \leftarrow 0$
5. **for** each vertex $u \in V[G]$
6. **do if** $color[u] =$ white
7. **then DFS-Visit(u)**

DFS-Visit(u)

1. $color[u] \leftarrow$ GRAY
2. $time \leftarrow time + 1$
3. $d[u] \leftarrow time$
4. **for** each $v \in Adj[u]$
5. **do if** $color[v] =$ WHITE
6. **then** $\pi[v] \leftarrow u$
7. DFS-Visit(v)
8. $color[u] \leftarrow$ BLACK
9. $f[u] \leftarrow time \leftarrow time + 1$

Running time is $\Theta(|V|+|E|)$

Example (DFS)



u
v
w
1/8
2/7
9/12
4/5
3/6
10/11
x
y
z
call stack
(x)
DFS-visit(y)
DFS-visit(v)
DFS-visit(u)
(z)
DFS-visit(w)
time

Parenthesis Theorem

Theorem

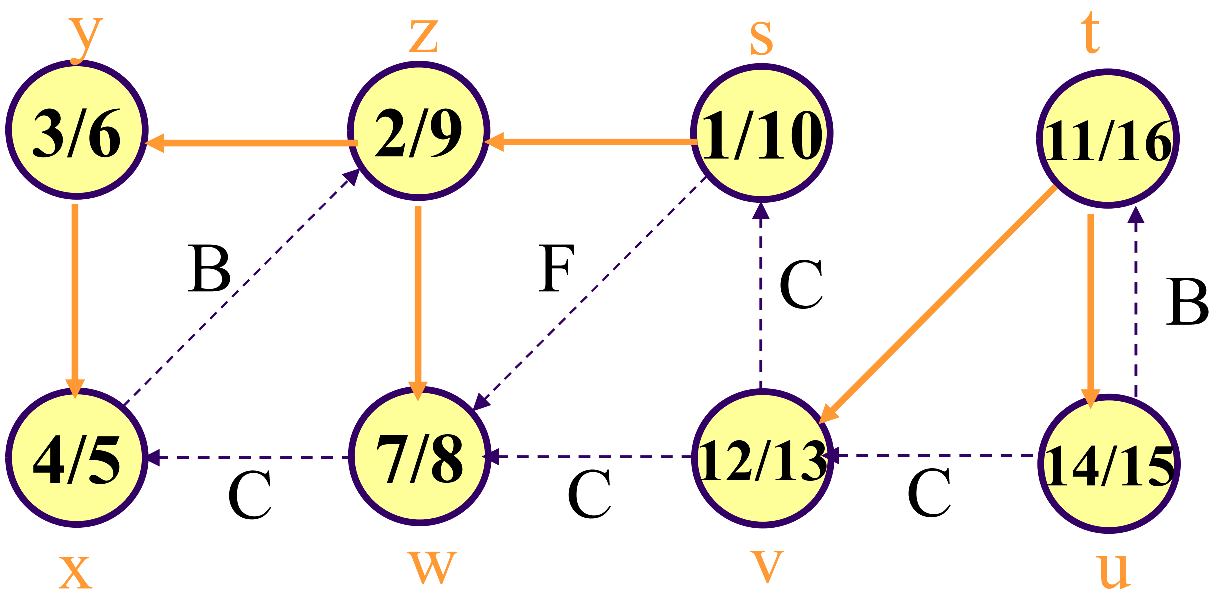
For all u, v, exactly one of the following holds:

1. $d[u] < f[u] < d[v] < f[v]$ or $d[v] < f[v] < d[u] < f[u]$ and neither u nor v is a descendant of the other.
2. $d[u] < d[v] < f[v] < f[u]$ and v is a descendant of u.
3. $d[v] < d[u] < f[u] < f[v]$ and u is a descendant of v.

- So $d[u] < d[v] < f[u] < f[v]$ *cannot* happen.
- *Corollary*

  v is a proper descendant of u if and only if $d[u] < d[v] < f[v] < f[u]$.

Another example



(s (z (y (x x) y) (w w) z) s) (t (v v) (u u) t)

White-path Theorem

Theorem

v is a descendant of u if and only if at time $d[u]$, there is a path $u \rightsquigarrow v$ consisting of only white vertices.

Proof

→ **direction:** Algorithm only proceeds on white vertices, so path must have been white.

← **direction:** By induction on the length of the white path. Let w be the vertex preceding v in the path. By induction assumption, w is a descendant of u. What color was v when we examined the edge (w, v)? White?... Gray/black?...

Classification of Edges

- ***Tree edge:*** Edges in G_π. v was found by exploring (u, v).
- ***Back edge:*** (u, v), where u is a descendant of v in G_π .
- ***Forward edge:*** (u, v), where v is a descendant of u, but not a tree edge.
- ***Cross edge:*** any other edge. Can go between vertices in same depth-first tree or in different depth-first trees.

Identification of Edges

- Edge type for edge (*u*, *v*) can be identified when it is first explored by DFS.
- Identification is based on the **color of *v***.
  - White – tree edge.
  - Gray – back edge.
  - Black – forward or cross edge.

$d[u] < d[v]$ (forward) $\qquad$ $d[u] > d[v]$ (cross)

Identification of Edges

Theorem:
In DFS of an undirected graph, we get only tree and back edges. No forward or cross edges.

Proof:

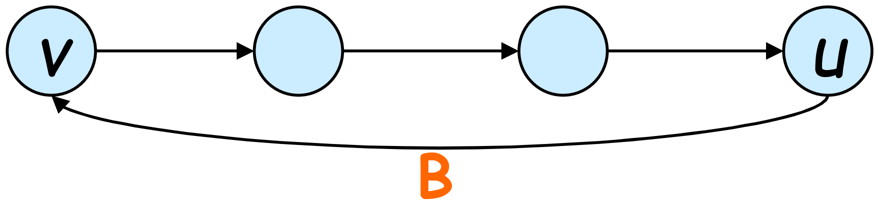
- Let $(u,v) \in E$. w.l.o.g let $d[u] < d[v]$.
  Then v must be discovered and finished before u is finished.
- If the edge (u,v) is explored first in the direction $u \rightarrow v$, then v is *white* until that time then it is a tree edge .
- If the edge is explored in the direction, $v \rightarrow u$, u is still gray at the time the edge is first explored, then it is a back edge.

Characterizing a DAG

Lemma
A directed graph G is acyclic iff a DFS of G yields no back edges.

Proof:

- $\Rightarrow$:
  - Suppose there is a back edge (u, v). Then v is an ancestor of u in depth-first forest.
  - Therefore, there is a path $v \rightsquigarrow u$, so $v \rightsquigarrow u \rightsquigarrow v$ is a cycle.

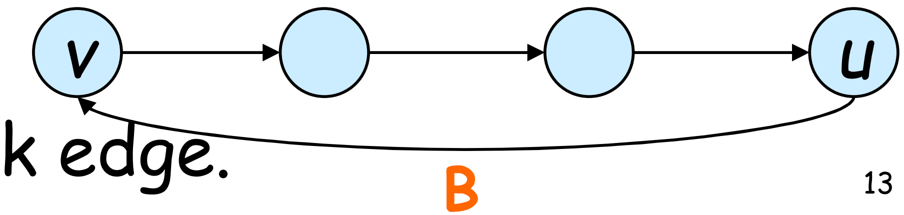


Characterizing a DAG

Lemma
A directed graph G is acyclic iff a DFS of G yields no back edges.

Proof (Cont.):

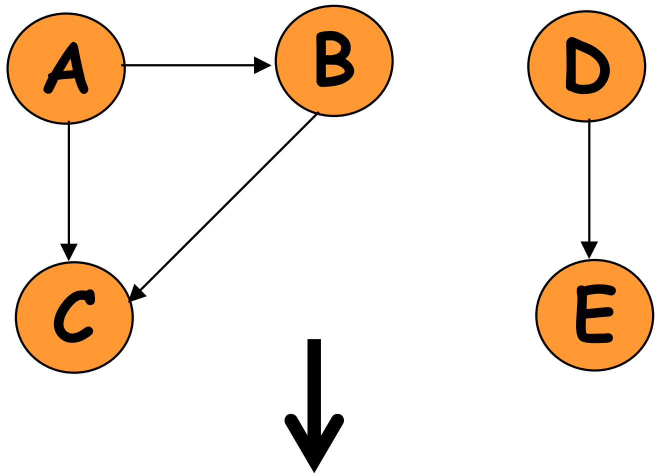
- $\Leftarrow$:
  - c : cycle in G, v : first vertex discovered in c, (u, v) : preceding edge in c.
  - At time $d[v]$, vertices of c form a white path $v \rightsquigarrow u$. Why?
  - By white-path theorem, u is a descendent of v in depth-first forest.
  - Therefore, (u, v) is a back edge.



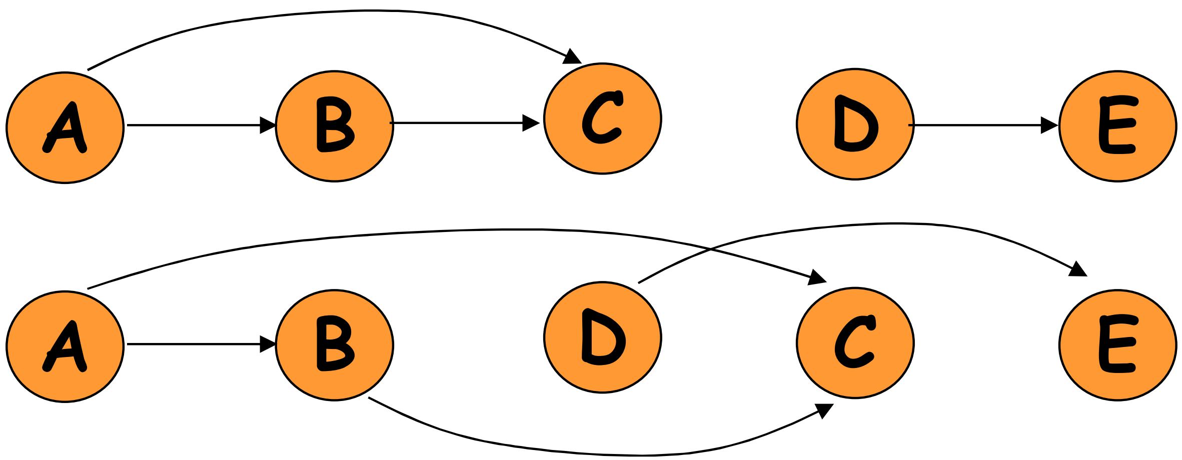
Topological Sort

Want to "sort" a directed acyclic graph (DAG).

Order the vertices such that all edges go forward



or



Topological Sort

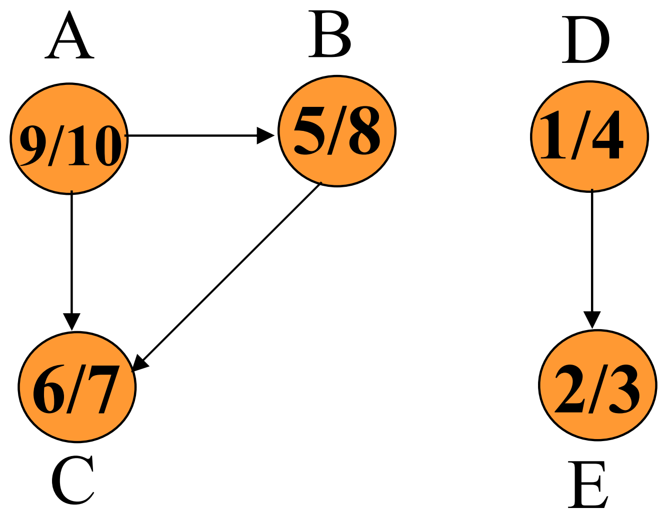
- Performed on a DAG.
- Linear ordering of the vertices of G such that if $(u, v) \in E$, then u appears somewhere before v.

<u>Topological-Sort (G)</u>

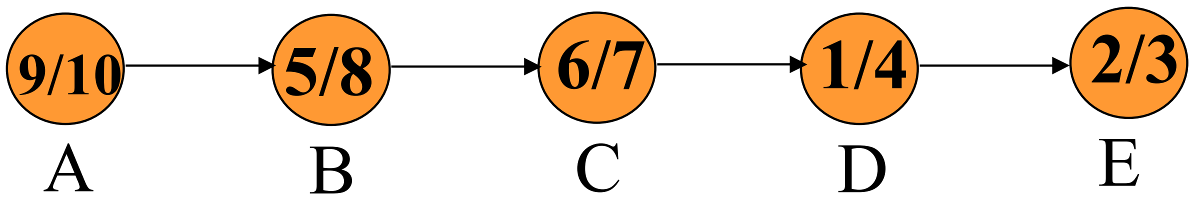
1. call DFS(G) to compute finishing times $f[v]$ for all $v \in V$
2. as each vertex is finished, insert it onto the front of a linked list
3. return the linked list of vertices

Running time is $\Theta(|V|+|E|)$

Example



Linked List:

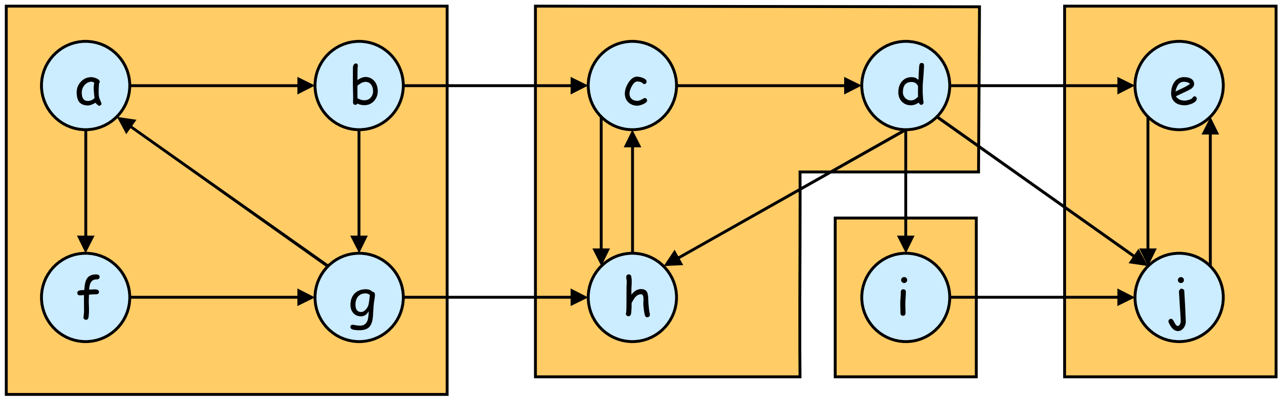


Correctness Proof

- Just need to show if $(u, v) \in E$, then $f[v] < f[u]$.
- When we explore (u, v), what are the colors of u and v?
  - u is gray.
  - Is v gray, too?
    - No, because then v would be an ancestor of u.
    - $\Rightarrow$ (u, v) is a back edge.
    - $\Rightarrow$ contradiction of Lemma (DAG has no back edges).
  - Is v white?
    - v is a descendant of u.
    - By parenthesis theorem, $d[u] < d[v] < \underline{f[v] < f[u]}$.
  - Is v black?
    - Then v is already finished.
    - Since we're exploring (u, v), we have not yet finished u.
    - $\Rightarrow$ $f[v] < f[u]$.

Strongly Connected Components

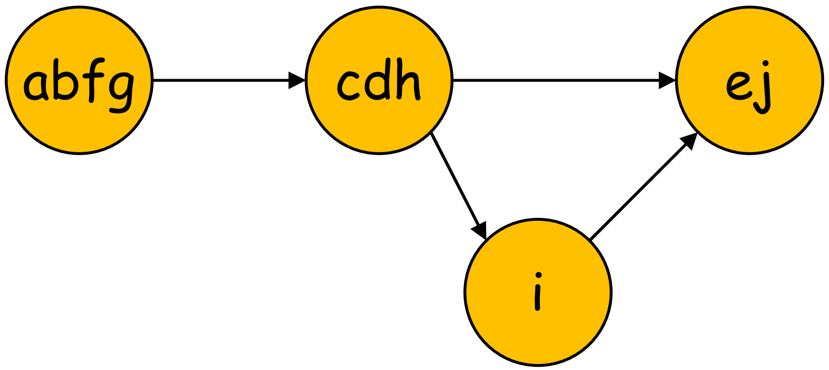
- G is strongly connected if every pair (u, v) of vertices in G is reachable from each other.
- A ***strongly connected component*** (***SCC***) of G is a maximal set of vertices $C \subseteq V$ such that for all $u, v \in C$, both $u \leadsto v$ and $v \leadsto u$ exist.



Component Graph

- $G^{SCC} = (V^{SCC}, E^{SCC})$.
- V^{SCC} has one vertex for each SCC in G.
- E^{SCC} has an edge if there is an edge between the corresponding SCC's in G.

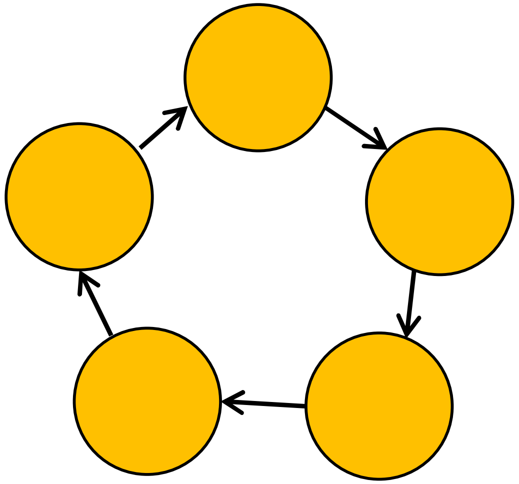
G^{SCC} for the example considered:



Claim: G^{SCC} is a DAG

Proof:

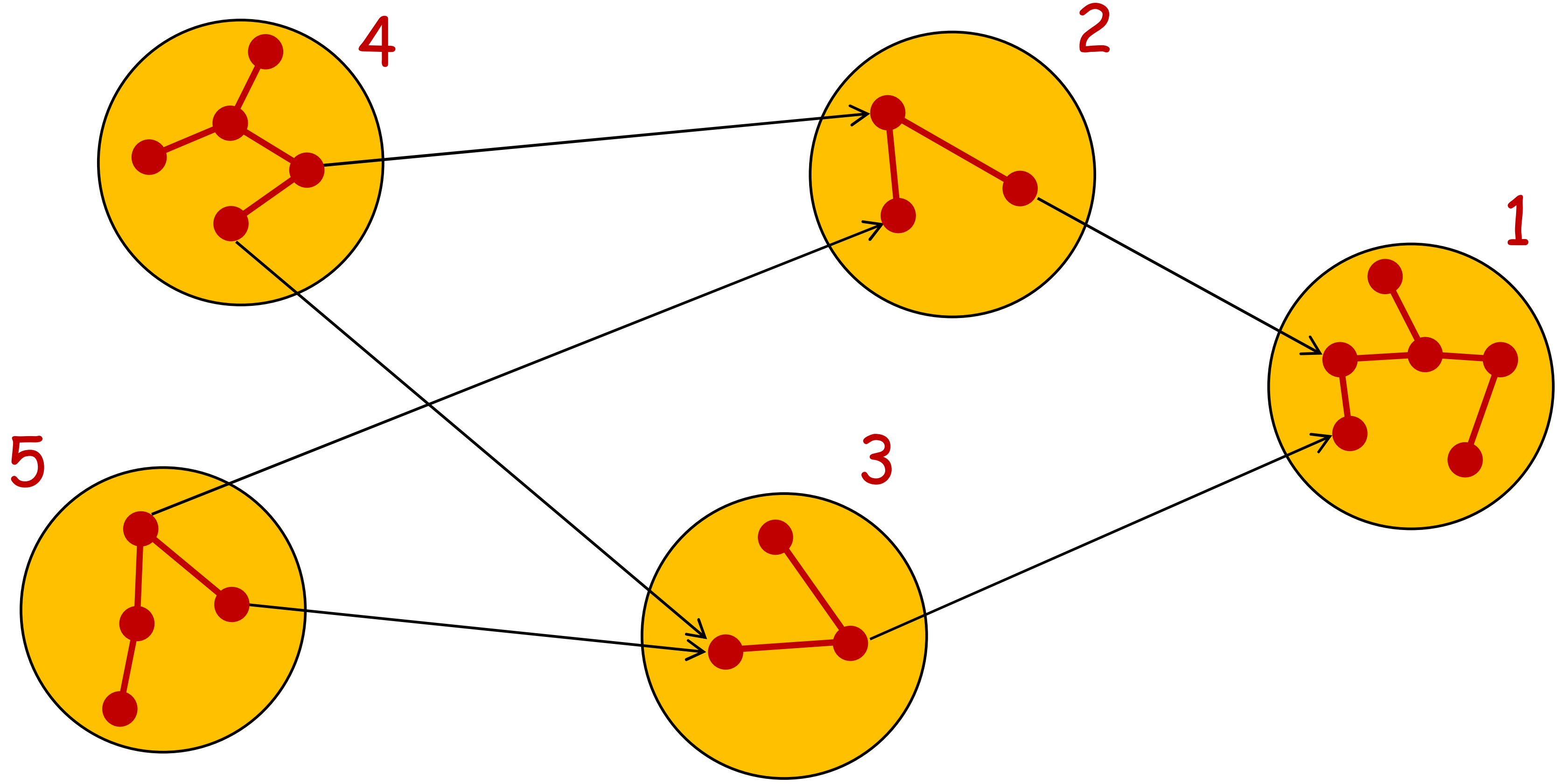
- Suppose for a contradiction that G^{SCC} contains a cycle



- All the vertices in these components are reachable from one another
- They should have been in the same component

Algorithm to determine SCCs

Idea: Do DFS, going "from the bottom up" in the Main Loop, so that each SCC becomes a separate tree



Algorithm to determine SCCs

Idea (cont.):

- We will do 2 rounds of DFS. The first round will give us the vertex order for the Main Loop of the second round

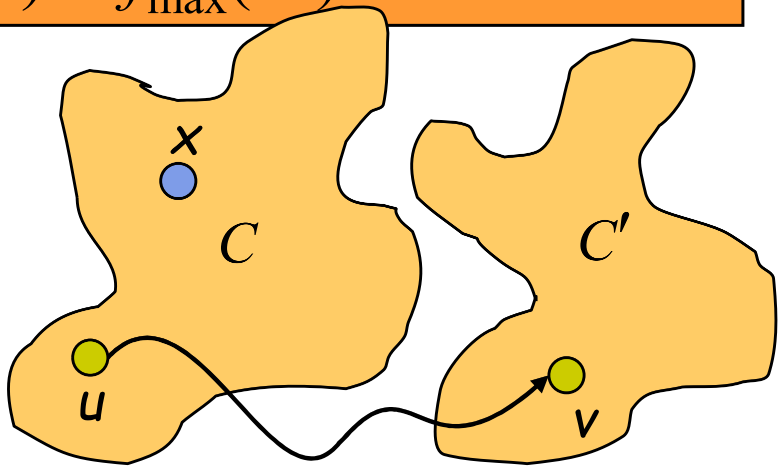
- In the first DFS, the Main Loop vertex order is arbitrary

Notation:

- $d[u]$ and $f\,[u]$ always refer to *first* DFS.
- Extend notation for d and f to sets of vertices $U \subseteq V$:
  - $d_{min}(U) = \min_{u \in U}\{d[u]\}$ (earliest discovery time)
  - $f_{max}(U) = \max_{u \in U}\{\,f\,[u]\}$ (latest finishing time)

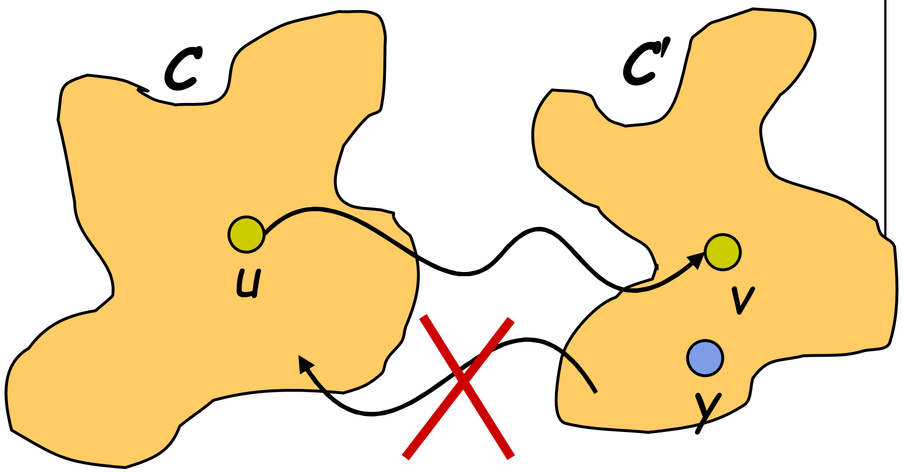
SCCs and DFS finishing times

Lemma
Let C and C' be distinct SCC's in $G = (V, E)$. Suppose there is a path $u \rightsquigarrow v$ such that $u \in C$ and $v \in C'$. Then $f_{max}(C) > f_{\max}(C')$.



Proof:

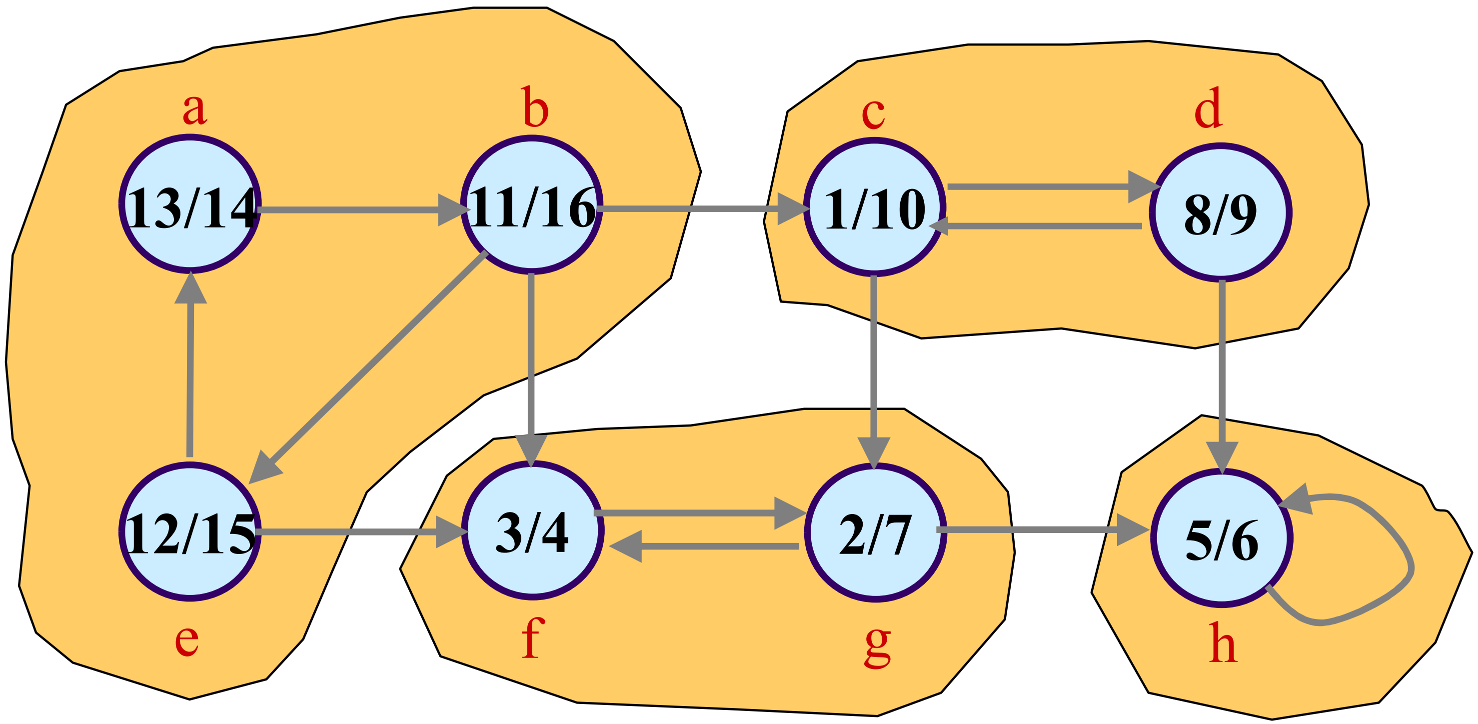
- Case 1: $d_{\min}(C) < d_{\min}(C')$
  - Let x be the first vertex discovered in C.
  - At time $d[x]$, all vertices in C and C' are white. Thus, there exist paths of white vertices from x to all vertices in C and C'.
  - By the white-path theorem, all vertices in C and C' are descendants of x in depth-first tree.
  - By the parenthesis theorem, $f[x] = f_{\max}(C) > f_{\max}(C')$.



- Case 2: $d_{\min}(C) > d_{\min}(C')$
  - Let y be the first vertex discovered in C'.
  - At time $d[y]$, all vertices in C' are white and there is a white path from y to each vertex in C' $\Rightarrow$ all vertices in C' become descendants of y. Again, $f[y] = f_{\max}(C')$.
  - At time $d[y]$, all vertices in C are also white.
  - By earlier lemma, we cannot have a path from C' to C.
  - So no vertex in C is reachable from y.
  - Therefore, at time $f[y]$, all vertices in C are still white.
  - Therefore, for all $w \in C$, $f[w] > f[y]$, which implies that $f_{\max}(C) > f_{\max}(C')$.

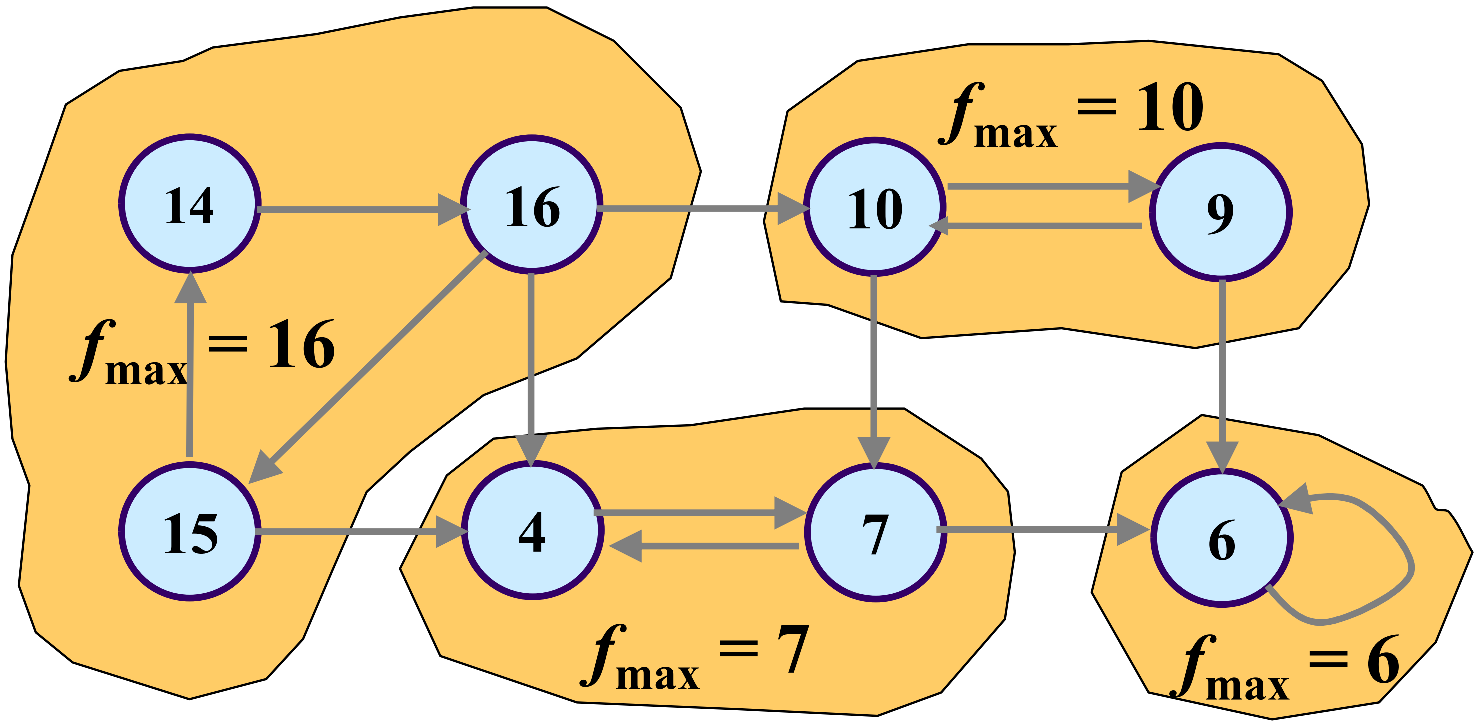
Example

First DFS



finishing times

This almost works: f_{max} gives opposite order
Solution: Reverse all the graph's edges for 2nd DFS



Transpose of a Directed Graph

- G^T = **transpose** of directed G.
  - $G^T = (V, E^T)$, $E^T = \{(u, v) : (v, u) \in E\}$.
  - G^T is G with all edges reversed.
- Can create G^T in $\Theta(V + E)$ time if using adjacency lists.
- G and G^T have the same SCC's. (u and v are reachable from each other in G if and only if reachable from each other in G^T.)

Algorithm to determine SCCs

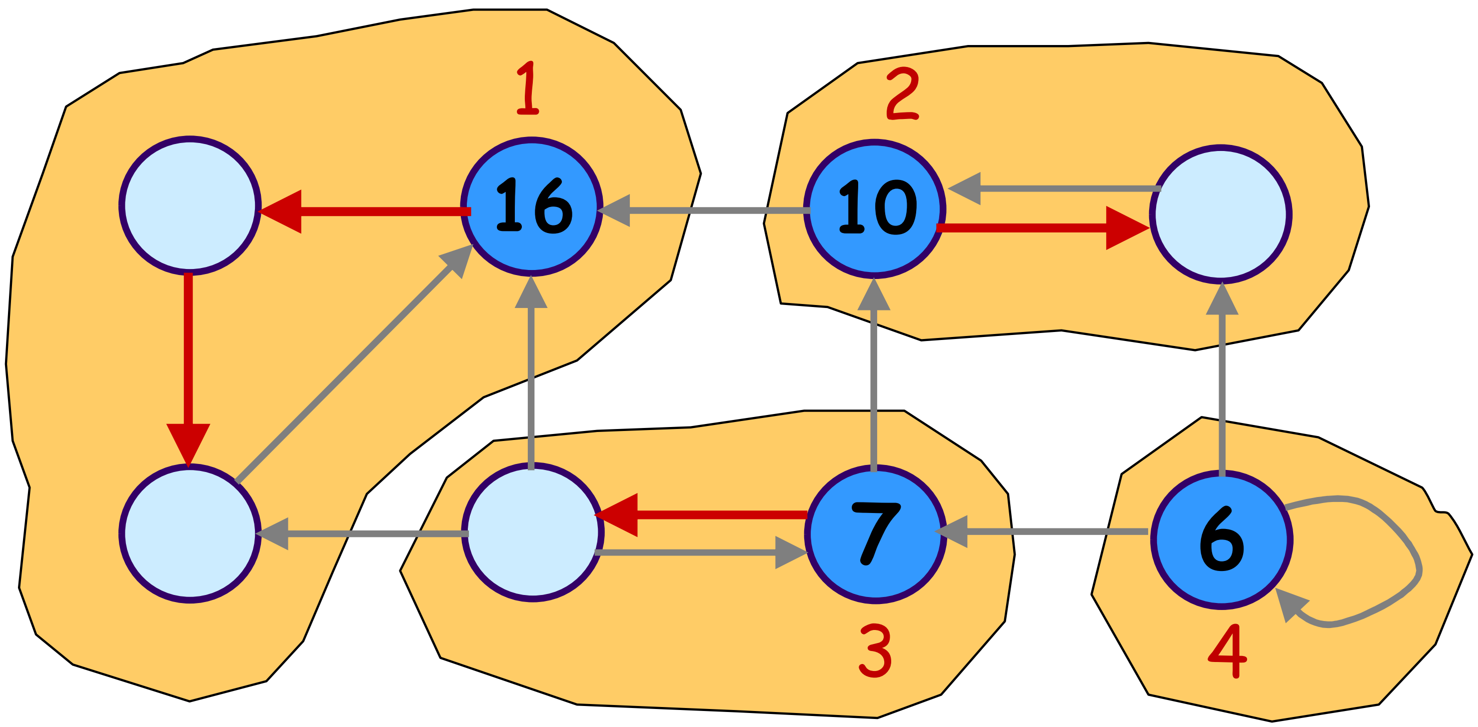
<u>SCC(G)</u>

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

Running time is $\theta(V+E)$

Example (continued)

Second DFS



Example