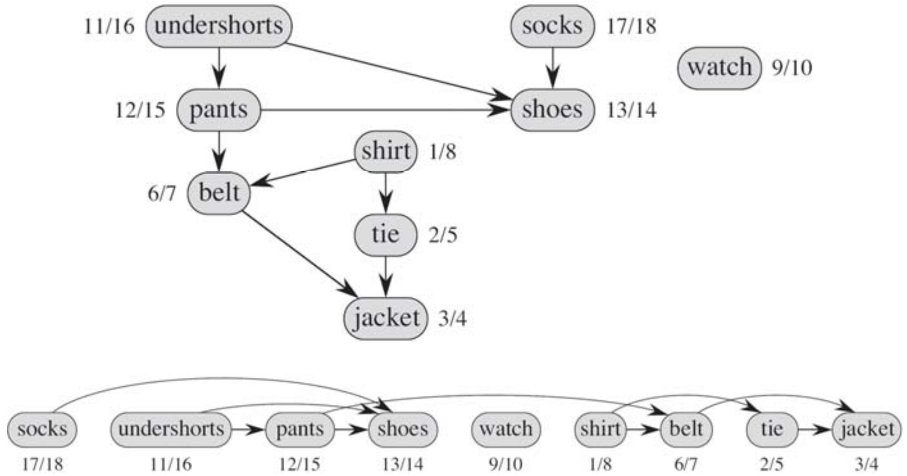
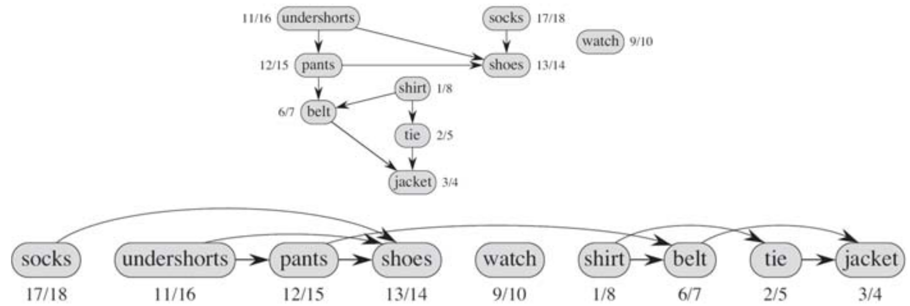
Topologically Sorted Dag



TOPOLOGICAL-SORT(G)

TOPOLOGICAL-SORT(G)

call DFS(G) to compute finishing times $v.f$ for each vertex v
as each vertex is finished, insert it onto the front of a linked list
return the linked list of vertices



Lemma and Theorem (1/5)

- **Lemma 11.** A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.

  Proof. $\Rightarrow$: Suppose that a depth-first search produces a back edge (u, v). Then vertex v is an ancestor of vertex u in the depth-first forest.

  Thus, G contains a path from v to u, and the back edge (u, v) completes a cycle.

  $\Leftarrow$: Suppose that G contains a cycle c. We show that a depth-first search of G yields a back edge.

Lemma and Theorem (2/5)

Let v be the first vertex to be discovered in c, and let (u, v) be the preceding edge in c.

At time $v.d$, the vertices of c form a path of white vertices from v to u.

By the white-path theorem, vertex u becomes a descendant of v in the depth-first forest.

➡ (u, v) is a back edge.

Lemma and Theorem (3/5)

- **Theorem 12.** TOPOLOGICAL-SORT produces a topological sort of the directed acyclic graph provided as its input.

  Proof. Suppose that DFS is run on a given dag $G = (V, E)$ to determine finishing times for its vertices.

  It suffices to show that for any pair of distinct vertices $u, v \in V$, if G contains an edge from u to v, then $v.f < u.f$.

Lemma and Theorem (4/5)

Consider any edge (u, v) explored by DFS(G).

When this edge is explored, v cannot be gray, since then v would be an ancestor of u and (u, v) would be a back edge, contradicting **Lemma 11**.

➡ v must be either white or black.

If v is white, it becomes a descendant of u, and so $v.f < u.f$.

Lemma and Theorem (5/5)

If v is black, it has already been finished, so that $v.f$ has already been set. Because we are still exploring from u, we have yet to assign a timestamp to $u.f$, and so once we do, we will have $v.f < u.f$ as well.

➡ For any edge (u, v) in the dag, we have $v.f < u.f$, proving the theorem.

Outline

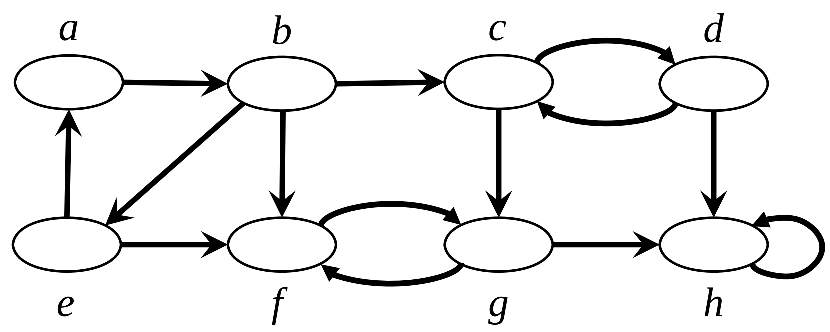
- Representations of Graphs
- Breadth-First Search
- Depth-First Search
- Topological Sort
- ***Strongly Connected Components***

Strongly Connected Components (1/3)

- We now consider a classic application of depth-first search: decomposing a directed graph into its strongly connected components.
- A strongly connected component of a directed graph $G = (V, E)$ is a maximal set of vertices $C \subseteq V$ such that for every pair of vertices u and v in C, we have both $u \rightsquigarrow v$ and $v \rightsquigarrow u$; that is, vertices u and v are reachable from each other.

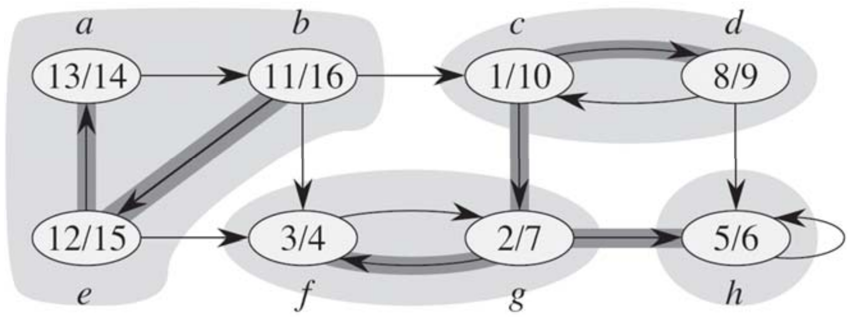
Strongly Connected Components (2/3)

- Find out strongly connected components of G:



Strongly Connected Components (3/3)

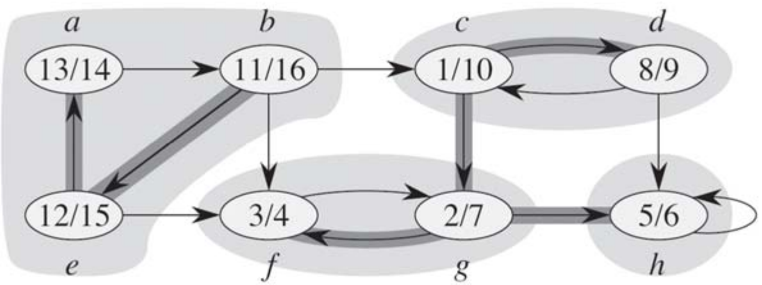
- Each shaded region is a strongly connected component of G.

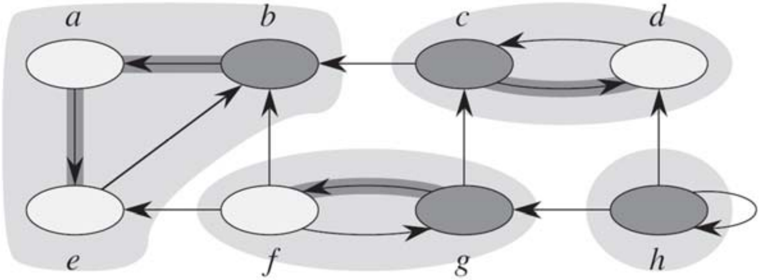


Find Strongly Connected Components

- Transpose G to be the graph $G^T = (V, E^T)$, where $E^T = \{(u, v) : (v, u) \in E\}$.

➡ It is interesting to observe that G and G^T have exactly the same strongly connected components: u and v are reachable from each other in G if and only if they are reachable from each other in G^T.





STRONGLY-CONNECTED-COMPONENTS

```
STRONGLY-CONNECTED-COMPONENTS(G)
1  call DFS(G) to compute finishing times u.f 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 u.f (as computed in line 1)
4  output the vertices of each tree in the depth-first forest formed in line 3 as a
       separate strongly connected component
```

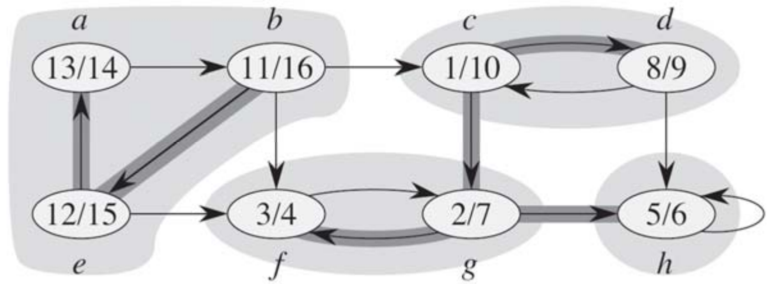
- The idea behind this algorithm comes from a key property of the component graph $G^{SCC} = (V^{SCC}, E^{SCC})$.

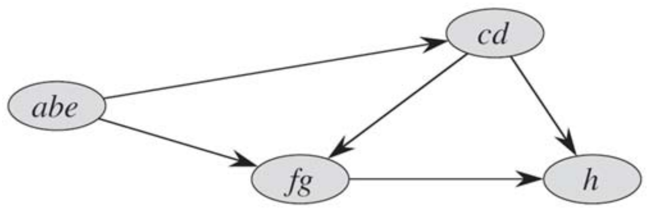
Component Graph (1/2)

- Suppose that G has strongly connected components $C_1, C_2, \ldots, C_k$.
- The vertex set V^{SCC} is $\{v_1, v_2, \ldots, v_k\}$, and it contains a vertex v_i for each strongly connected component C_i of G.
- There is an edge $(v_i, v_j) \in E^{SCC}$ if G contains a directed edge (x, y) for some $x \in C_i$ and some $y \in C_j$.

➡ Key property: the component graph is a dag.

Component Graph (2/2)





Lemmas and Theorem (1/16)

- **Lemma 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 G contains a path $u \leadsto u'$. Then G cannot also contain a path $v' \leadsto v$.

  Proof. If G contains a path $v' \leadsto v$, then it contains paths $u \leadsto u' \leadsto v'$ and $v' \leadsto v \leadsto u$.

➡ u and v' are reachable from each other.

➡ Contradicting the assumption that C and C' are distinct strongly connected components.

Lemmas and Theorem (2/16)

- We shall see that by considering vertices in the second depth-first search in decreasing order of the finishing times that were computed in the first depth-first search, we are, in essence, visiting the vertices of the component graph (each of which corresponds to a strongly connected component of G) in topologically sorted order.

Lemmas and Theorem (3/16)

- We extend the notation for discovery and finishing times to sets of vertices.
- If $U \subseteq V$, then we define $d(U) = \min_{u \in U} \{u.d\}$ and $f(U) = \max_{u \in U} \{u.f\}$.
- That is, $d(U)$ and $f(U)$ are the earliest discovery time and latest finishing time, respectively, of any vertex in U.

Lemmas and Theorem (4/16)

- **Lemma 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 $f(C) > f(C')$.

  Proof. We consider two cases, depending on which strongly connected component, C or C', had the first discovered vertex during the depth-first search.

  If $d(C) < d(C')$, let x be the first vertex discovered in C. At time $x.d$, all vertices in C and C' are white.

Lemmas and Theorem (5/16)

At that time, G contains a path from x to each vertex in C consisting only of white vertices.

Because $(u, v) \in E$, for any vertex $w \in C'$, there is also a path in G at time $x.d$ from x to w consisting only of white vertices: $x \rightsquigarrow u \to v \rightsquigarrow w$.

By the white-path theorem, all vertices in C and C' become descendants of x in the depth-first tree.

By **Corollary 8**, x has the latest finishing time of any of its descendants, and so $x.f = f(C) > f(C')$.

Lemmas and Theorem (6/16)

If $d(C) < d(C')$, let y be the first vertex discovered in C'. At time $y.d$, all vertices in C' are white and G contains a path from y to each vertex in C' consisting only of white vertices.

By the white-path theorem, all vertices in C' become descendants of y in the depth-first tree, and by **Corollary 8**, $y.f = f(C')$.

At time $y.d$, all vertices in C are white. Since there is an edge (u, v) from C to C', **Lemma 13** implies that there cannot be a path from C' to C.

Lemmas and Theorem (7/16)

➡ No vertex in C is reachable from y.

At time $y.f$, all vertices in C are still white.

➡ For any vertex $w \in C$, we have $w.f > y.f$, which implies that $f(C) > f(C')$.

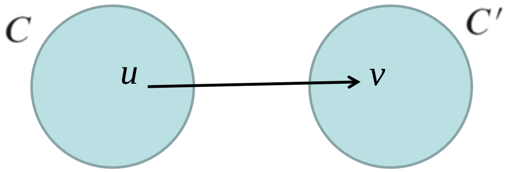
Lemmas and Theorem (8/16)

- **Corollary 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 $f(C) < f(C')$.

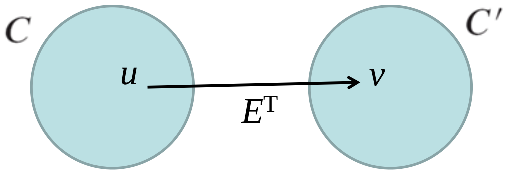
  Proof. Since $(u, v) \in E^T$, we have $(v, u) \in E$.

  Because the strongly connected components of G and G^T are the same, **Lemma 14** implies that $f(C) < f(C')$.

Lemmas and Theorem (9/16)



Lemma 14 $f(C) > f(C')$



Corollary 15 $f(C) < f(C')$

Lemmas and Theorem (10/16)

- Let us examine what happens when we perform the second depth-first search, which is on G^{T}.
- We start with the strongly connected component C whose finishing time $f(C)$ is maximum.
- The search starts from some vertex $x \in C$, and it visits all vertices in C.
- By **Corollary 15**, G^{T} contains no edges from C to any other strongly connected component, and so the search from x will not visit vertices in any other component.

Lemmas and Theorem (11/16)

➡ The tree rooted at x contains exactly the vertices of C.

- Having completed visiting all vertices in C, the search in line 3 selects as a root a vertex from some other strongly connected component C' whose finishing time $f(C')$ is maximum over all components other than C.
- Again, the search will visit all vertices in C', the only edges in G^{T} from C' to any other component must be to C, which we have already visited.

Lemmas and Theorem (12/16)

➡ In general, when the depth-first search of G^{T} in line 3 visits any strongly connected component, any edges out of that component must be to components that the search already visited.

➡ Each depth-first tree will be exactly one strongly connected component.