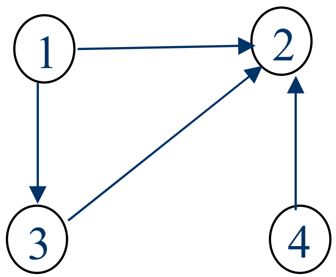
Lecture 4
Applications of Depth-First Search

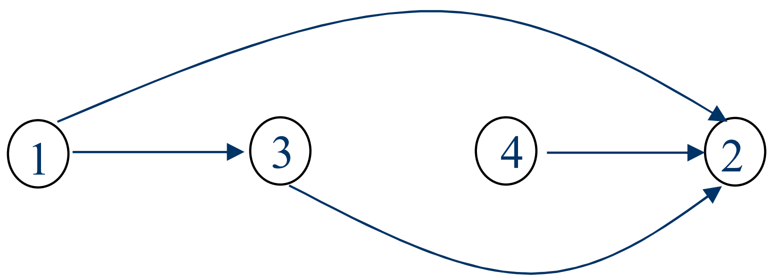
- Topological Sort
  -- (for directed acyclic graph)

- Strongly Connected Components Decomposing
  -- (for directed graph)

The first application: Topological Sort

- Directed acyclic graph (abbreviated as dag) are used in many applications to indicate precedences among events.
- A topological sort of a directed acyclic graph $G = (V, E)$ 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.

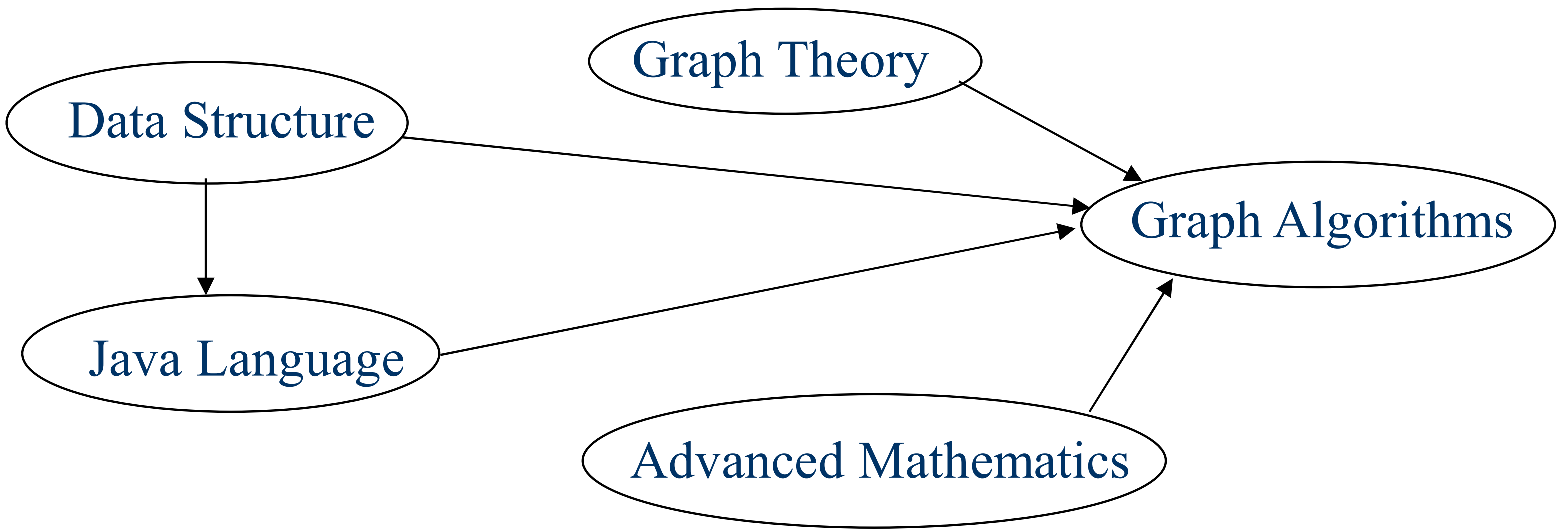




How about if G contains a cycle?

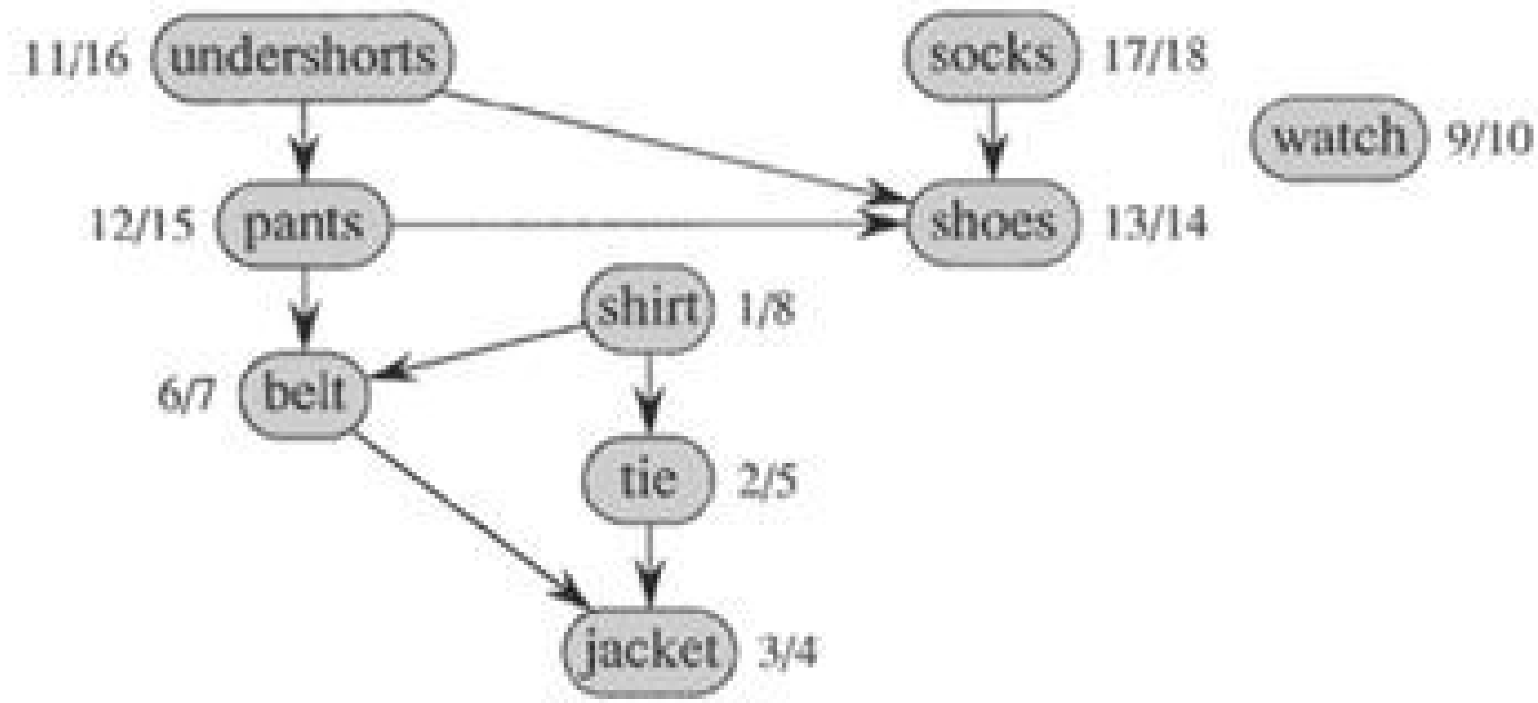
Applications of Topological Sort

- Courses arrangement in schools
- Genome rearrangement
- A general life-related application—dressing order



Can you give the topological order?

Application--Dressing Up



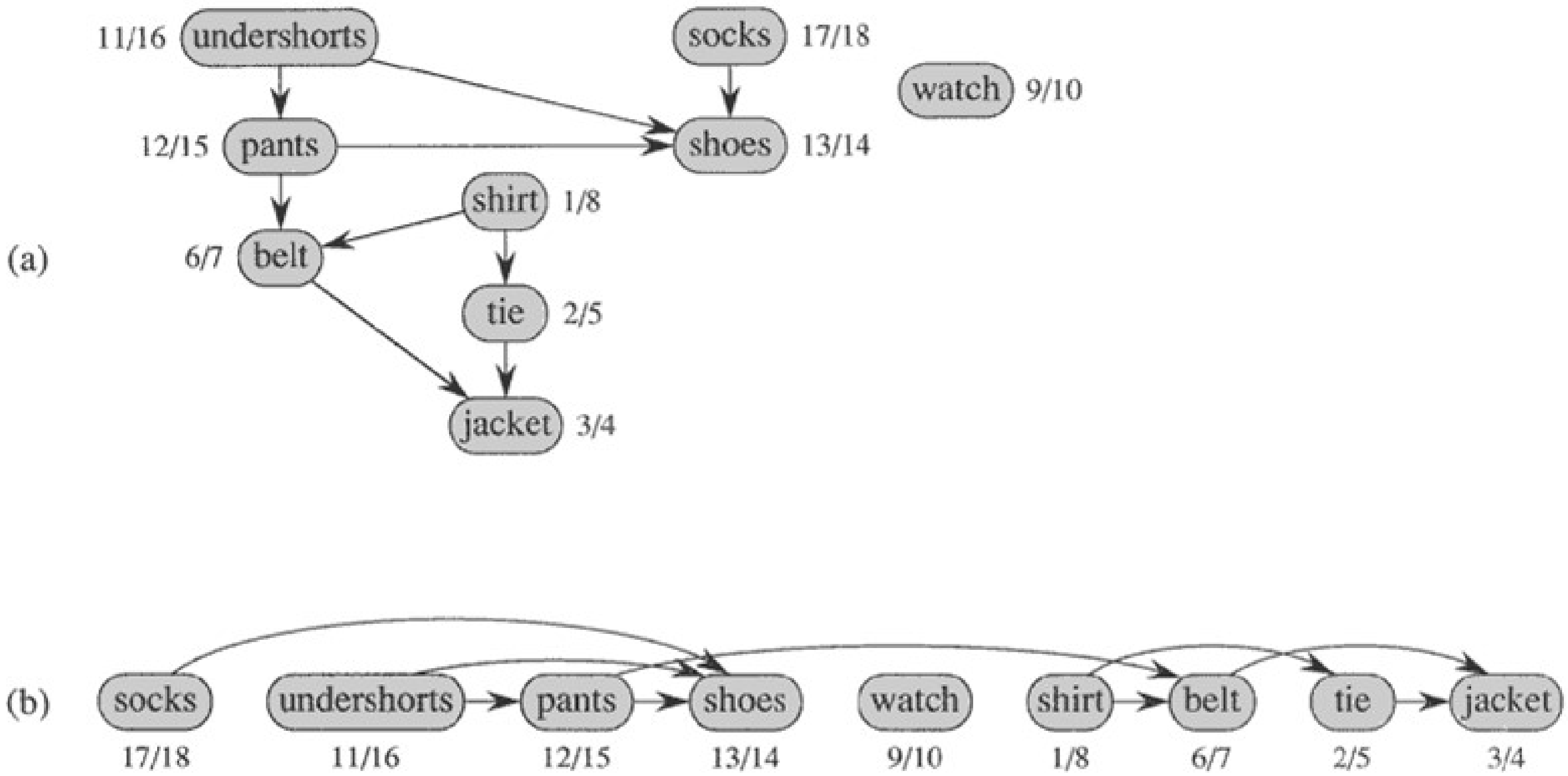
Topological Sort Problem

- The problem
  - Input: A directed acyclic graph $G = (V, E)$.
  - Output: A topological order of all the vertices of G.

The Algorithm

- Topological-Sort（G）
  - 1，call DFS(G) to compute finishing times *v.f* for each vertex *v*.
  - 2，as each vertex is finished, insert it onto the front of a linked list
  - 3，return the linked list of vertices.

Application--Dressing Up



Time Complexity

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

- Time Complexity Analysis:
  - Line 1: $O(V + E)$
  - Line 2: $O(V)$
  - Line 3: $O(1)$
  - Total: $O(V + E)$

A Property for DAG

- Lemma 22.11： A directed graph G is acyclic if and only if a depth-first search of G yields no back edges.
- Proof:
  - =>: (Proof by contradiction) Suppose that there is a back edge (u, v). Then, vertex v is an ancestor of vertex u in the depth-first forest. Since there is also a path from v to u in G, the back edge (u, v) completes a cycle. Hence, a contradiction.
  - <=: (Proof by contradiction) Suppose that G contains a cycle c. 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 in G. By the white-path theorem, vertex u becomes a descendant of v in the depth-first forest. Therefore, (u, v) is a back edge. Hence, a contradiction.

Correctness Proof of the Algorithm

- Theorem 22.12: TOPOLOGICAL-SORT(G) produces a topological sort of a directed acyclic graph G.
- Proof: Suppose that DFS is run on a given dag $G = (V, E)$. It suffices to show that if (u, v) is an edge in G, then $u.f > v.f$.
- By lemma 22.11, (u, v) cannot be a back edge, therefore, there are only three cases:
  - (u, v) is a tree edge, u is ancestor of v, $u.f > v.f$;
  - (u, v) is a forward edge, u is ancestor of v, $u.f > v.f$;
  - (u, v) is a cross edge, when u is still being processed, v is black, $u.f > v.f$;
- In all the three cases, we have $u.f > v.f$, so the proof is done.
- A question: how about a back edge?

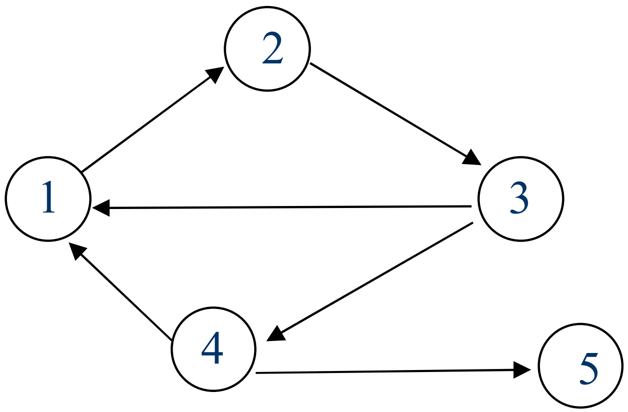
Another Method for Topological Sort

- Repeat:
  - Select a vertex v whose in-degree is zero, append it to the end of a list.
  - Delete v and all edges leaving it.
- Return the list.
- Proof sketch:
  - for any $(u,v) \in E$, the in-degree of v can not be zero before deleting u, which implies u appears in front of v.

Exercise 22.4-5

Strongly Connected Components Decomposing

- A strongly connected component of a directed graph $G = (V, E)$ is a vertex induced sub-graph $G'=(V', E')$ of G, such that :
  - (1) Every pair of vertices in V' are reachable from each other in G';
  - (2) Any other sub-graph that contains more vertices than G' **does not** satisfy (1).



Is {1,2,3} a strongly connected component?

Is {1,2,3,4} a strongly connected component?

And {5}?

The Strongly Connected Components Decomposition Problem

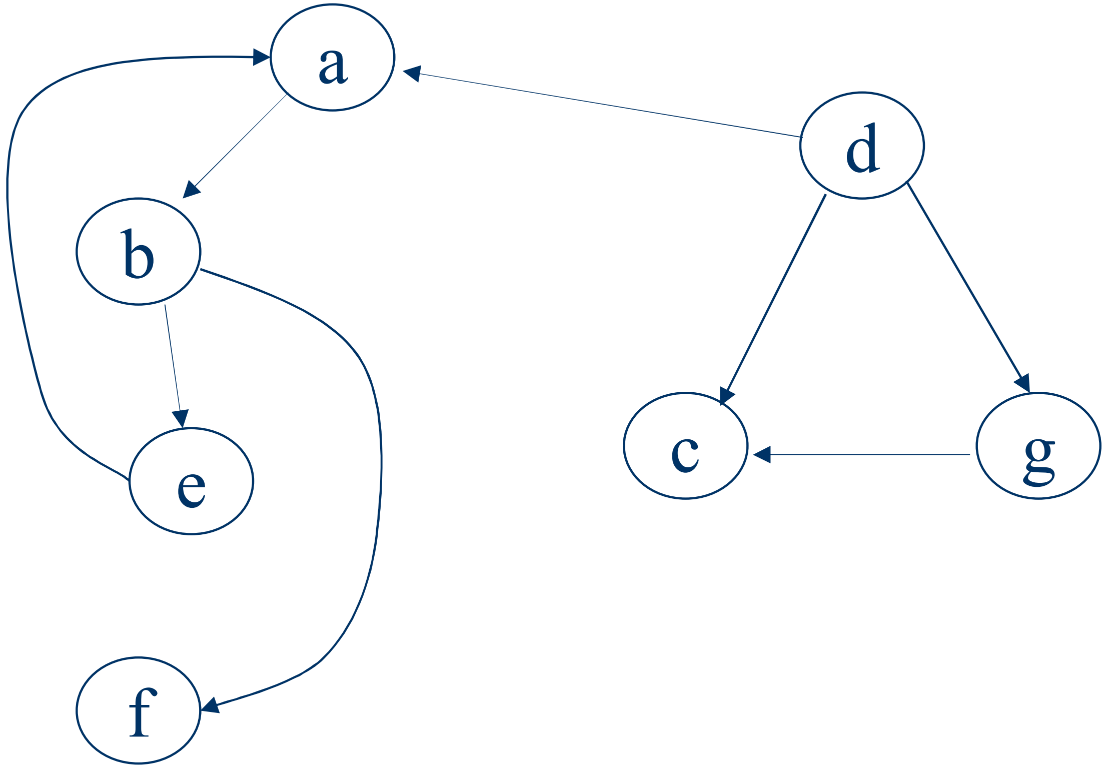
- The problem:
  - Input: A directed graph $G = (V, E)$.
  - Output: All the strongly connected components of G.

Notes:

Sometimes, **Strongly Connected Component** is abbreviated as **SCC**, while **Strongly Connected Components** is abbreviated as **SCCs**.

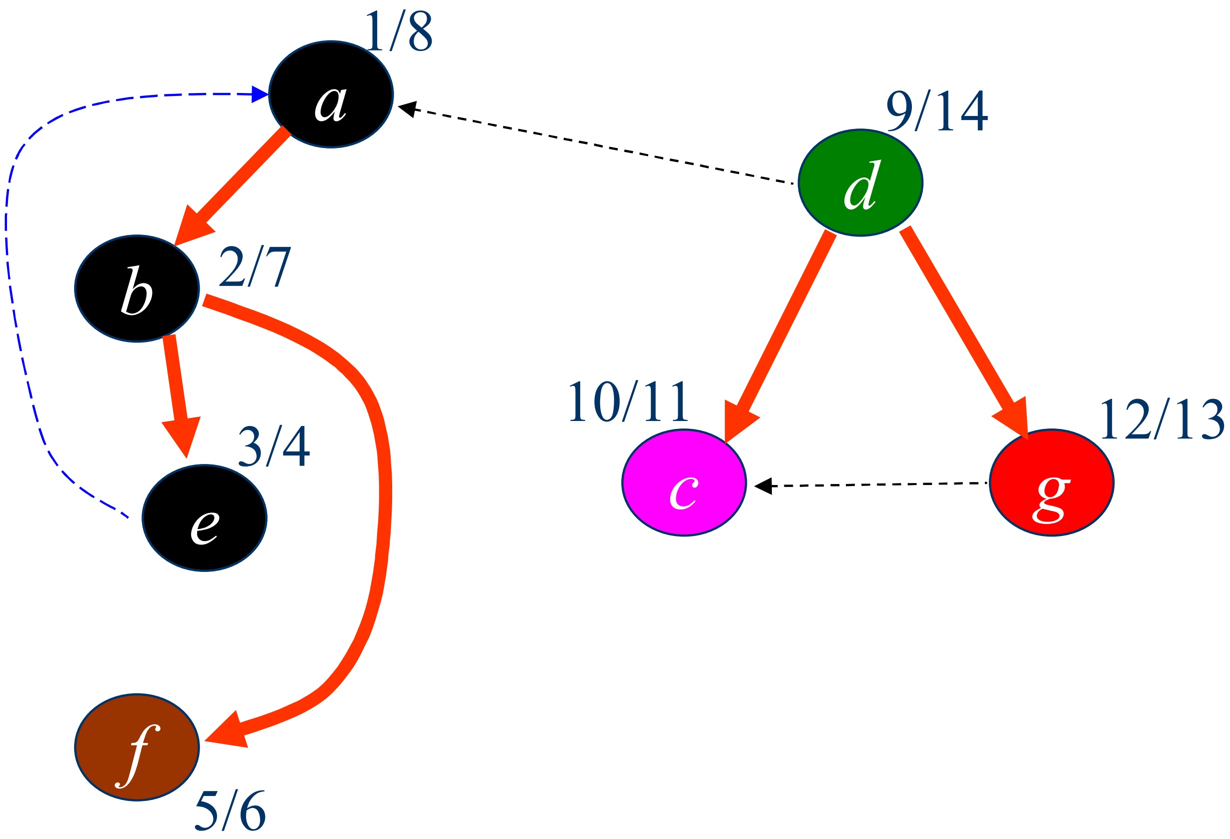
An Example used to illustrate the procedure of SCCs Decomposing based on DFS algorithm

- Give all the strongly connected components of the following graph.
- Give the DFS forest of the following directed graph.



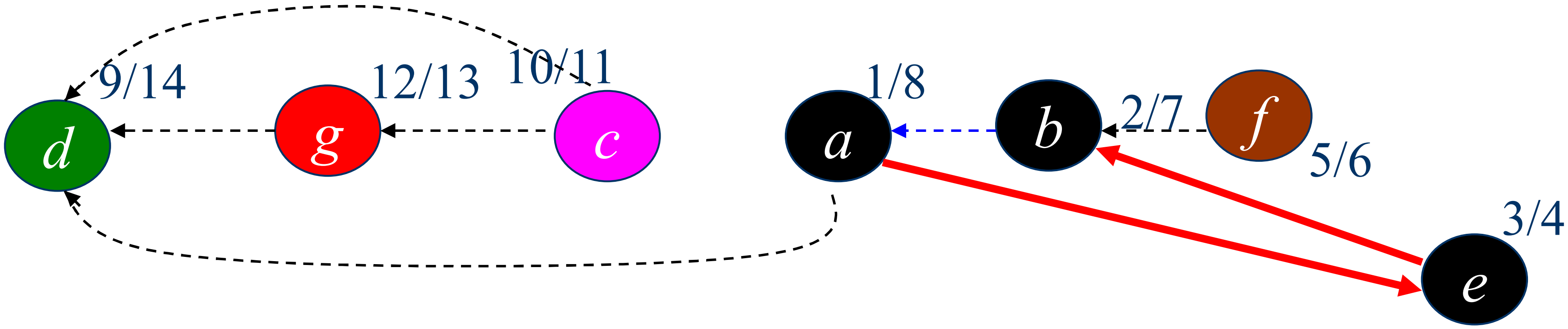
Observations

- For each tree in a depth-first forest of a graph G:
  - It contains **all** the vertices in the same SCC.
  - But, it may also contain some that are **not**!



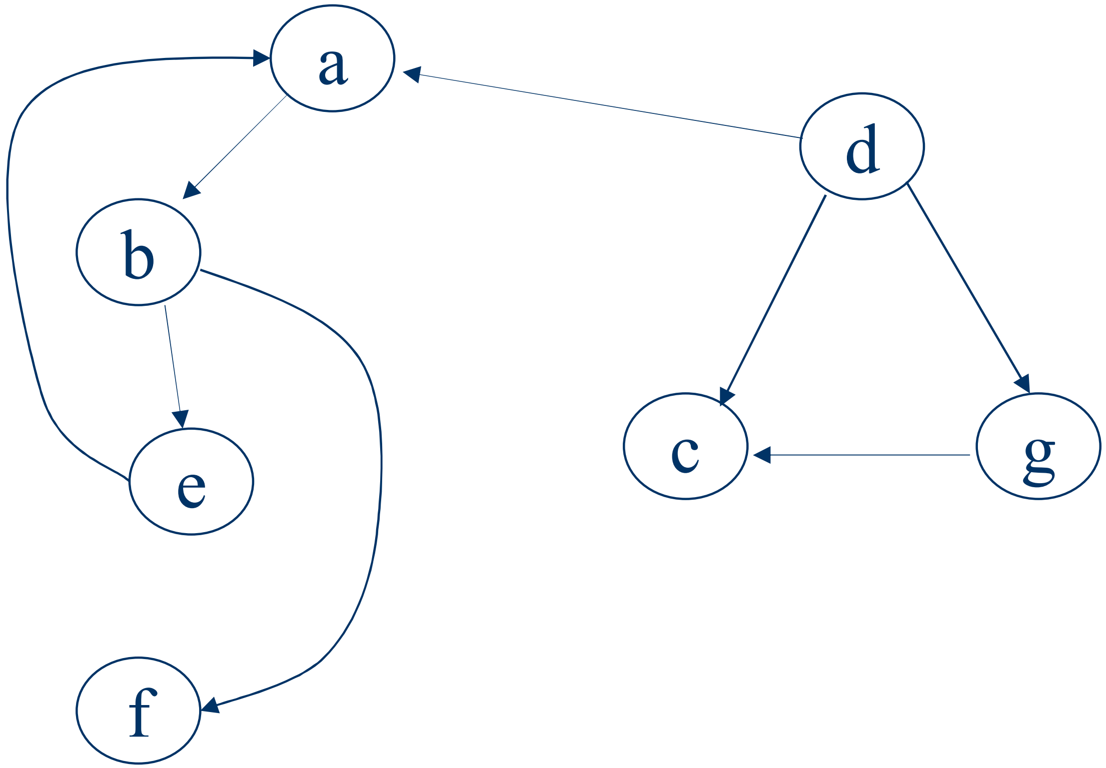
- Observation: **Running DFS algorithm on the original graph alone, we can't find all SCCs correctly.**

- What happens if we **reverse the edges** of the original graph and depth-first search the new graph in **decreasing** f order?



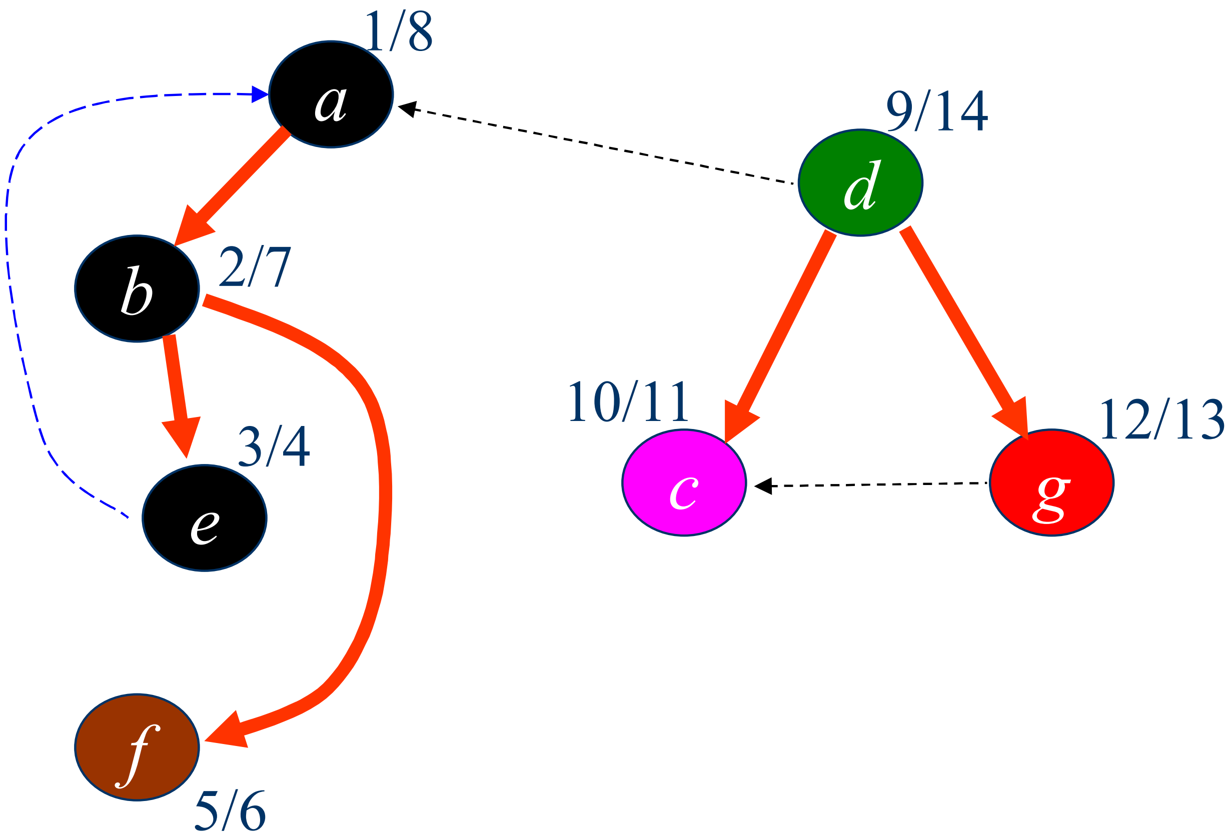
An Example used to illustrate the procedure of SCCs Decomposing based on DFS algorithm

- Give all the strongly connected components of the following graph.
- Give the DFS forest of the following directed graph.



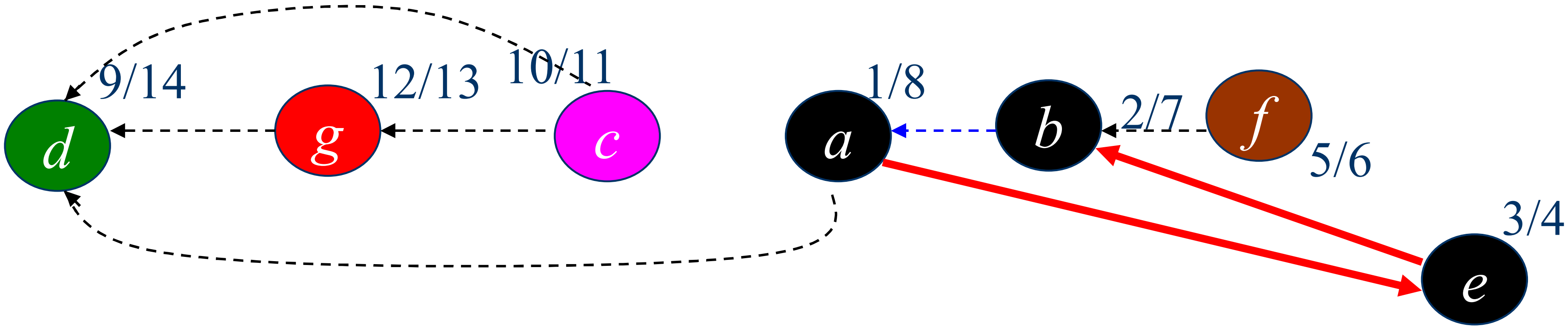
Observations

- For each tree in a depth-first forest of a graph G:
  - It contains **all** the vertices in the same SCC.
  - But, it may also contain some that are **not**!



- Observation: **Running DFS algorithm on the original graph alone, we can't find all SCCs correctly.**

- What happens if we **reverse the edges** of the original graph and depth-first search the new graph in **decreasing** *f* order?



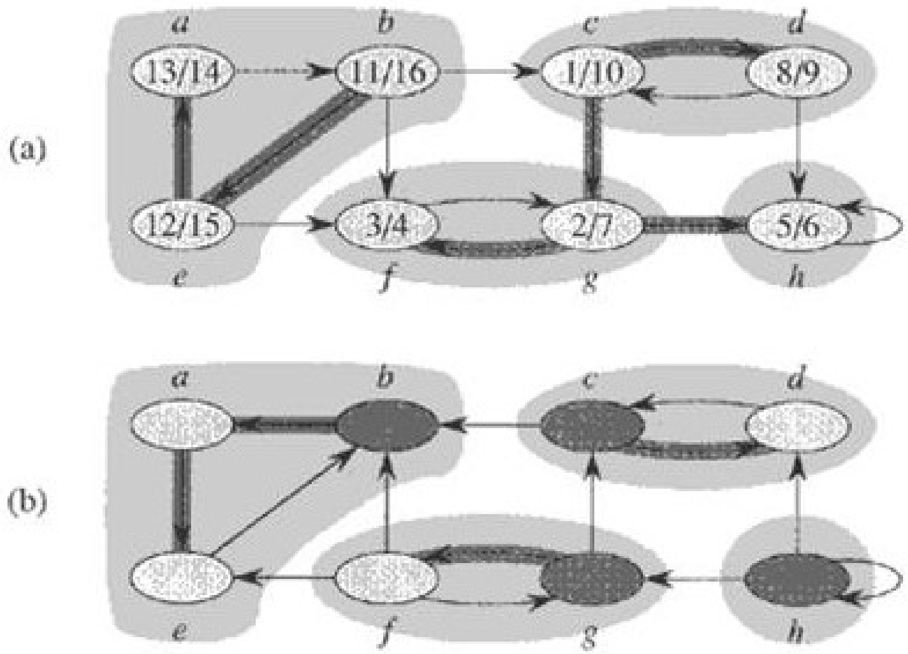
Transpose of a directed graph G

- Definition:

  Given a directed graph $G = (V, E)$, the transpose of G is the graph $G^T=(V, E^T)$, where $E^T=\{(v, u) | (u, v)\in E\}$.

- Given an adjacency-list representation of G, the time to create G^T is $O(V + E)$. Why?

An Example



(a): The graph G with its SCCs shaded

(b): The transpose G^T of G with SCCs shaded

Relationship Between G and G^{T}

- Given a directed graph $G = (V, E)$
  - G and G^{T} have exactly the same strongly connected components, or in other words,
  - u and v are reachable from each other in G if and only if they are reachable from each other in G^{T}.

The Algorithm

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

Time Complexity

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

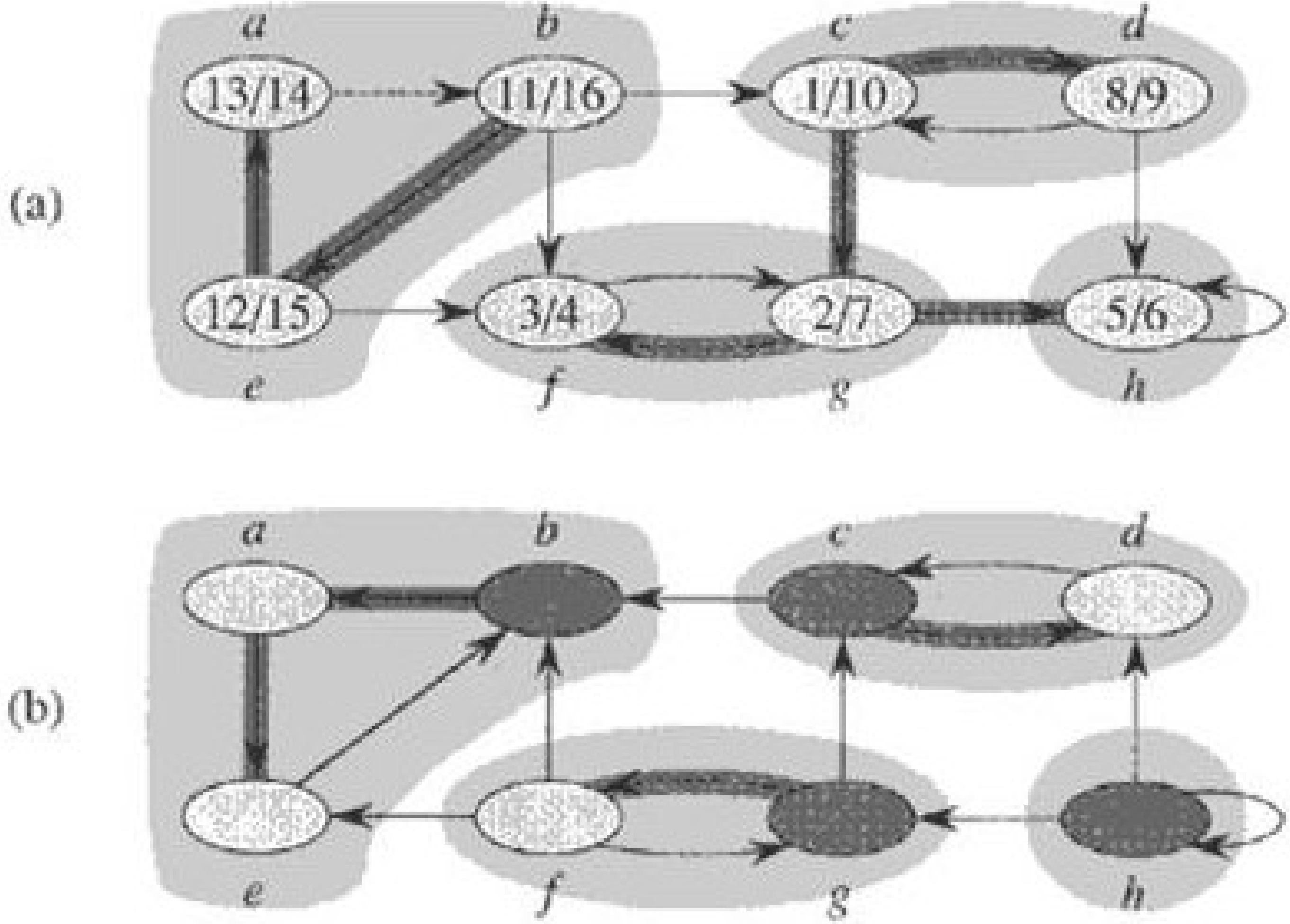
Time Complexity:

line 1, line 2, line 3: $\Theta(V + E)$

line 4: $\Theta(V)$

total: $\Theta(V + E)$

An example



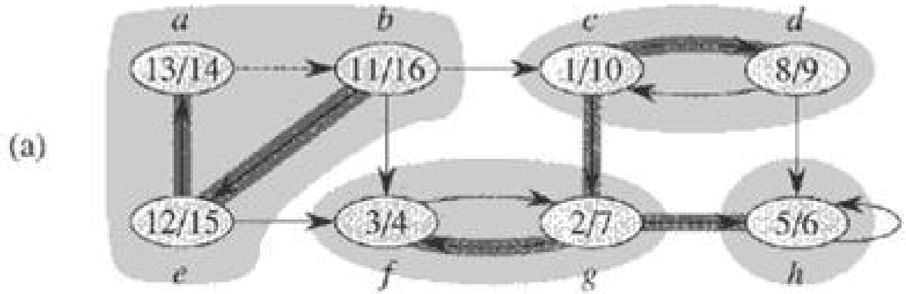
Preparations – Part 1
The Component Graph

- The component graph $G^{SCC} = (V^{SCC}, E^{SCC})$ of a directed graph $G = (V, E)$ is defined as follows:
  - Suppose that G has strongly connected components C_1, C_2, …, C_k
  - the vertex set $V^{SCC} = \{v_i \mid v_i$ corresponds to component C_i of G, $i = 1,2, \ldots, k\}$
  - the edge set $E^{SCC} = \{(v_i, v_j) \mid G$ contains a directed edge (x, y) for some $x \in C_i$ and some $y \in C_j$, $i, j = 1, 2, \ldots, k, i \neq j\}$
- Another way: Contracting edge with vertices in the same strongly connected component.

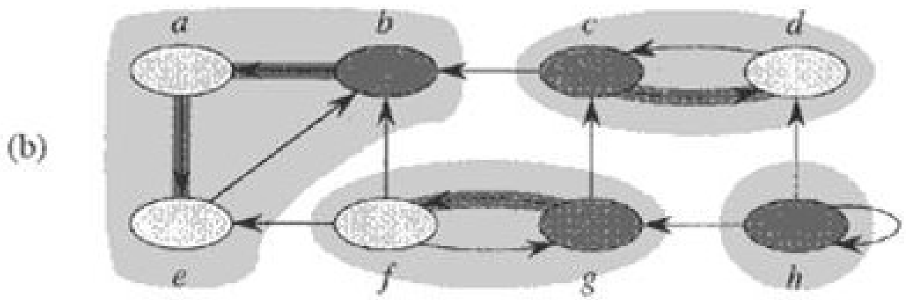
Preparations – Part 1
A Key Property of The Component Graph

- Lemma 22.13 :The component graph $G^{SCC} = (V^{SCC}, E^{SCC})$ is a directed acyclic graph.
- Proof:(by contradiction)
  - Suppose that G^{SCC} is cyclic, that is, there exist two vertices v_1, $v_2 \in V^{SCC}$ such that v_1 and v_2 are reachable from each other.
  - Since v_1 and v_2 represent the two strongly connected components C_1 and C_2 of G respectively.
  - Then according to the definition of component graph, vertices in C_1 and C_2 are reachable from each other, which contradicts the definition of strongly connected component.

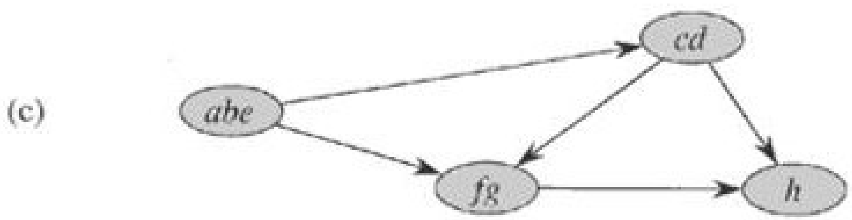
Preparations – Part 1
An Example



(a): The graph G with its SCCs shaded



(b): The transpose G^T of G with SCCs shaded



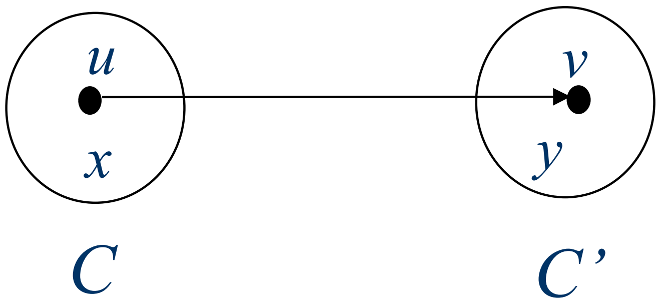
(c): The component graph $G^{SCC} = (V^{SCC}, E^{SCC})$ of G

Preparations – Part 2
Relationship between the SCCs and the f[]'s

- Extension of the notation for $d[\cdot]$ and $f[\cdot]$ to sets of vertices
- If $U \subseteq V$, define
  - $d(U) = \min_{u \in U}\{d[u]\}$, the discovery time of vertex set U, that is, the earliest discovery time of any vertex in U;
  - $f(U) = \max_{u \in U}\{f[u]\}$, the finishing time of vertex set U, that is, the latest finishing time of any vertex in U.

Preparations – Part 2
Relationship between the SCCs and the f[]'s

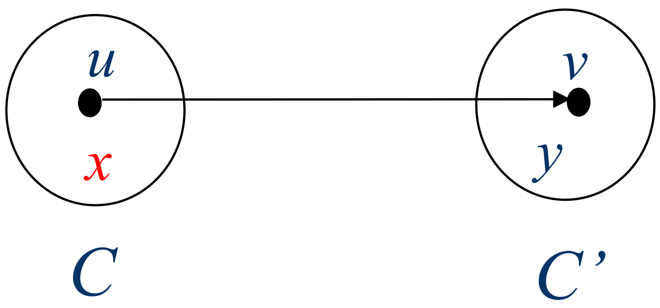
- 引理22.14: 设C 和 C' 是两个强连通分支，如果存在边$(u, v) \in E$, 其中 $u \in C$，$v \in C'$，则 $f(C) > f(C')$。
- 证明：分两种情况
  - 第一次DFS先扫描的C中的某个顶点，$d(C) < d(C')$
  - 第一次DFS先扫描的C'中的某个顶点，$d(C') < d(C)$



Preparations – Part 2

Proof of Lemma 22.14 (continued)

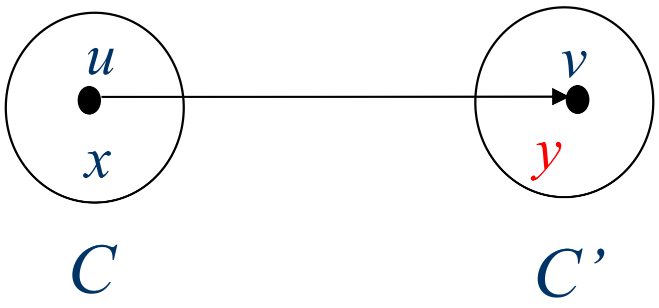
- Case 1: $d(C) < d(C')$
  - 设x是C中第一个被扫描的顶点，则$d[x]$时刻，C和C'中的所有顶点都是白色的。x到C中的每个顶点都存在一条由白色顶点构成的路径；同时，边(u,v)的存在，使得x到C'中的每个顶点w也有一条白色路径:$x \sim u \rightarrow v \sim w$，根据白色路径定理, C和C'中的所有顶点都是u的后代，由于后代的区间都是祖先区间的子区间，所以$f[x]=f(C)>f(C')$.



Preparations – Part 2

Proof of Lemma 22.14 (continued)

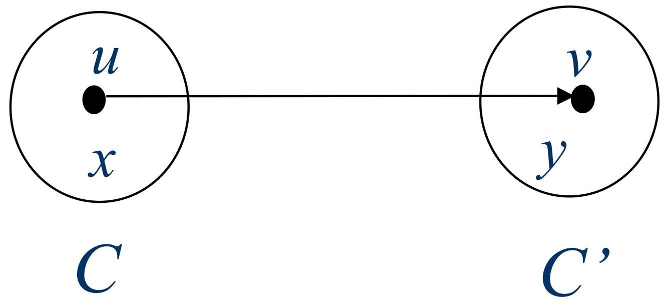
- Case 2: $d(C')< d(C)$
  - 设y是C'中第一个被扫描的顶点，则$d[y]$时刻，C'中的所有顶点都是白色的。x到C'中的每个顶点都存在一条由白色顶点构成的路径，根据白色路径定理C'中的所有顶点都是u的后代，由于后代的区间都是祖先区间的子区间，所以$f[y]=f(C')$.
  - $d[y]$时刻，C中的所有顶点都是白色的。因为 (u, v) 是从 C 到 C'的边，引理22.13保证，不存在从C' 到 C 的路径，所以y不存在任何路径通向C中的顶点，所以，$f[y]$时刻，C中的所有顶点还是白色的。因此对任意的$w\in C$，$f[w]>f[y]$，即$f(C) > f(C')$.



Preparations – Part 2
Relationship between the SCCs and the f[]'s

- Lemma 22.14 Let C and C' be two 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:



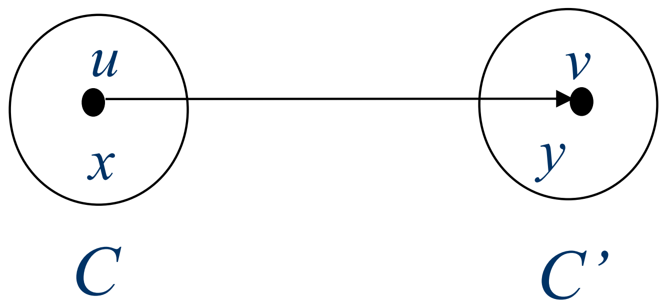
Case 1: $d(C) < d(C')$

Case 2: $d(C) > d(C')$

Preparations – Part 2
Proof of Lemma 22.14 (continued)

- Case 1: $d[C] < d[C']$

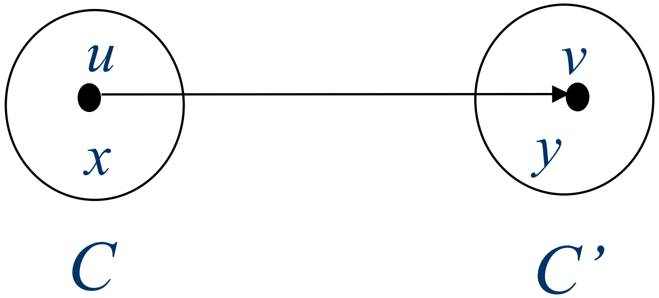
  Let x be the first vertex discovered in C. At time $d[x]$, all vertices in C and C' are white. There is a path in G from x to each vertex in C consisting only of white vertices. Because $(u, v) \in E$, for any $w \in C'$, there is also a path from x to w in G consisting only of white vertices. By the white-path theorem, all vertices in C and C' become descendants of x in the depth-first tree. By Corollary 22.8, $f[x]=f(C)>f(C')$.



Preparations – Part 2
Proof of Lemma 22.14 (continued)

- Case 2: $d[C] > d[C']$

  Let y be the first vertex discovered in C'. At time $d[y]$, all vertices in C' are white and there is a path in G from y to each vertex in C' consisting only of white vertices. By white-path theorem, all vertices in C' become descendants of y in the depth-first tree, and by Corollary 22.8, $f[y]=f(C')$. At time $d[y]$, all vertices in C are white. Since there is an edge (u, v) from C to C', Lemma 22.13 implies that there cannot be a path from C' to C. Hence, no vertex in C is reachable from y. At time $f[y]$, therefore, all vertices in C are still white. Thus, for any vertex $w \in C$, we have $f[w]>f[y]$, which implies that $f(C) > f(C')$.



Preparations – Part 2
A Corollary

- 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 $f(C) < f(C')$.
- E^T中的边都是从先结束(f[]小)的到后结束的(f[]大) 。
  - Proof: $(u, v) \in E^T \Rightarrow (v, u) \in E$, G and G' has the same strongly connected components, Lemma 22.14 implies $f(C) < f(C')$.
- Note: Here, the finishing times are got from the first depth-first search

Final Destination
Correctness Proof of the Algorithm

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

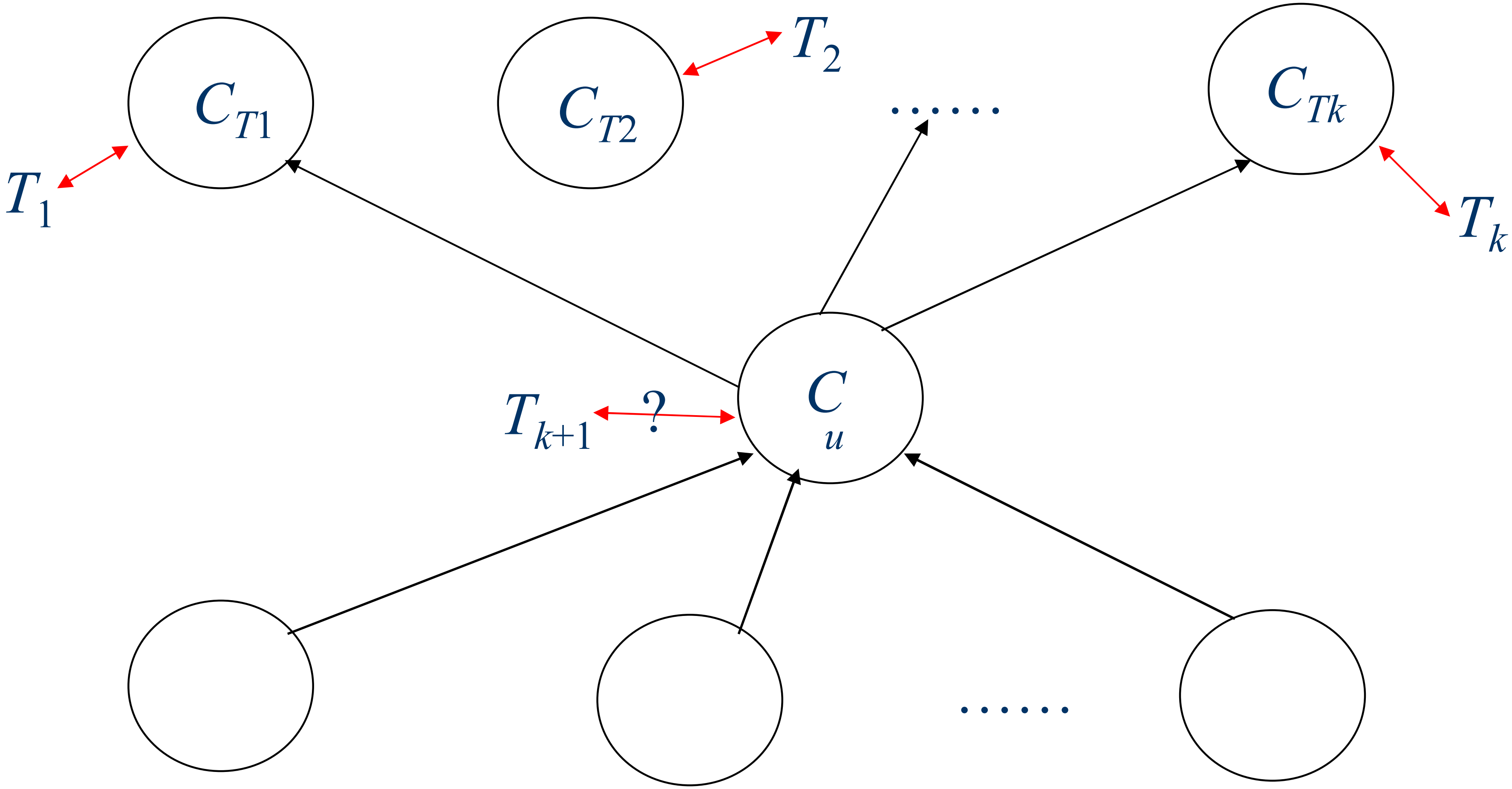
Proof

- 我们证明对G^T进行DFS得到的每一棵树都对应一个强连同分支。对找到的树的序号k进行归纳证明。
- 当k=1时，设u是该树的根，对应的强连同分支为C，$f[u]$是所有顶点结束时间最大的，$f[u]=f(C)$；$d[u]$时刻，C中的顶点都是白色的，根据白色路径定理，C中的所有顶点都是u的后代，从而都在以u为根的树中。
- 根据推论22.15，C中顶点不可能有出边到C之外的顶点，从而u不存在到任何其它连通分支中顶点的路径。所以，以u为根的树中，包含且只包含C中的所有顶点。
- 归纳假设：前k棵树都对应强连通分支

Proof Continued

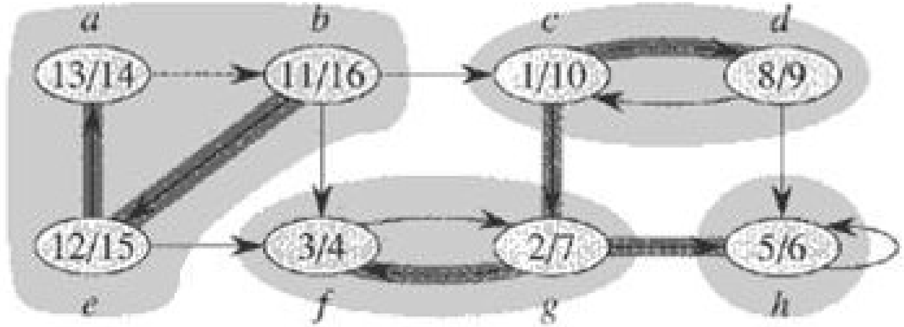
- 则找到第k+1棵树时，设u是该树的根，对应的强连同分支为C，$f[u]$是未扫描的顶点中结束时间最大的，$f[u]=f(C)>f(C')$, C'是任意尚未找到的连通分支；此时C中的顶点都是白色的，根据白色路径定理，C中的所有顶点都是u的后代，从而都在以u为根的树中。
- 根据推论22.15，在 G^T中，C中顶点只可能有出边到C之外的已经找到的强连通分支中的顶点，从而u不存在到任何其它未找到的连通分支中顶点的路径，在对G^T进行DFS时，所有未找到的强连通分支中的顶点都不是u的后代。所以，以u为根的树中，包含且只包含C中的所有顶点。

Proof (Continued)

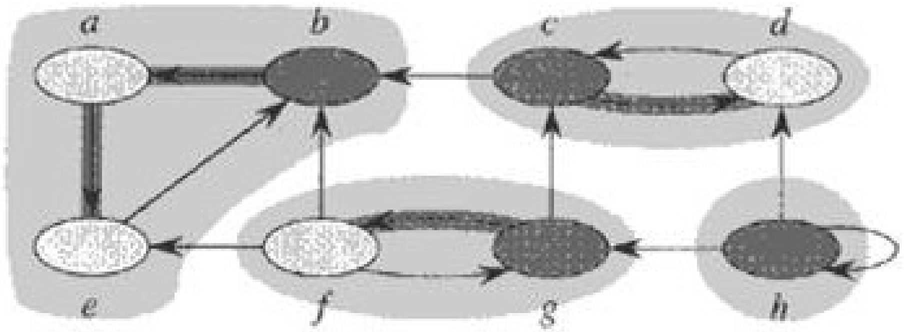


Review the Example

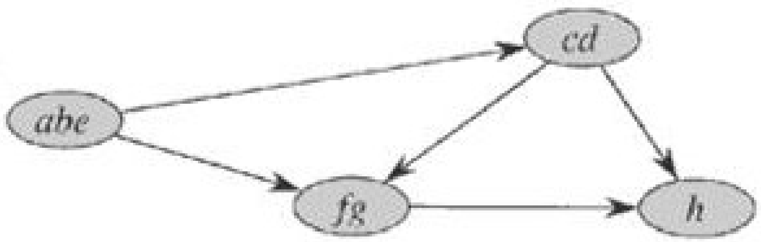
(a)



(b)



(c)



Proof (continued)

Proof: we will prove that **the vertices of each tree form a strongly connected component**, by induction on the number k of depth-first trees found in the depth-first search of G^T in line 3.

1. Basic step: $k = 0$, it is trivially true.
2. Inductive hypothesis: the first k trees produced in line 3 are strongly connected components.
3. Inductive step: Consider the $(k+1)$st tree produced.
   - Let the root of the $(k+1)$st tree be vertex u, and let u be in strongly connected component C. By induction hypothesis, no vertices in C had been searched before, and since u has the biggest $f[]$ among all the remaining vertices in G^T, $f[u] = f[C] > f[C']$ for any strongly connected component C' other than C that has yet not been visited.
   - At time $d[u]$, all other vertices of C are white. By the white-path theorem, **all other vertices of C are descendants of u in its depth-first tree**.
   - Moreover, by induction hypothesis, Corollary 22.15 and $f[C] > f[C']$ got above, any edge in G^T that leaves C must point to SCCs that have already been visited. Thus, **no vertex in any SCC other than C will be a descendant of u** during the depth-first search of G^T.

Thus, the vertices of the depth-first tree in G^T that is rooted at u form exactly one strongly connected component

Conclusion

- Topological Sort algorithm
  -- (for directed acyclic graph)
  - A key property for DAG
  - Proof of the correctness of the algorithm
- Strongly Connected Components Decomposing Algorithm-- (for directed graph)
  - A key property of the component graph
  - The relationship between the SCCs and the f[]'s
  - Proof of the correctness of the algorithm

Homework

- 22.4-2
- 22.4-5
- 22.5-5
- 22.5-7
- Probem 22-3