Chapter 22: Elementary Graph Algorithms IV

About this lecture

- Review of Strongly Connected Components (SCC) in a directed graph

- Finding all SCC

  (i.e., decompose a directed graph into SCC)

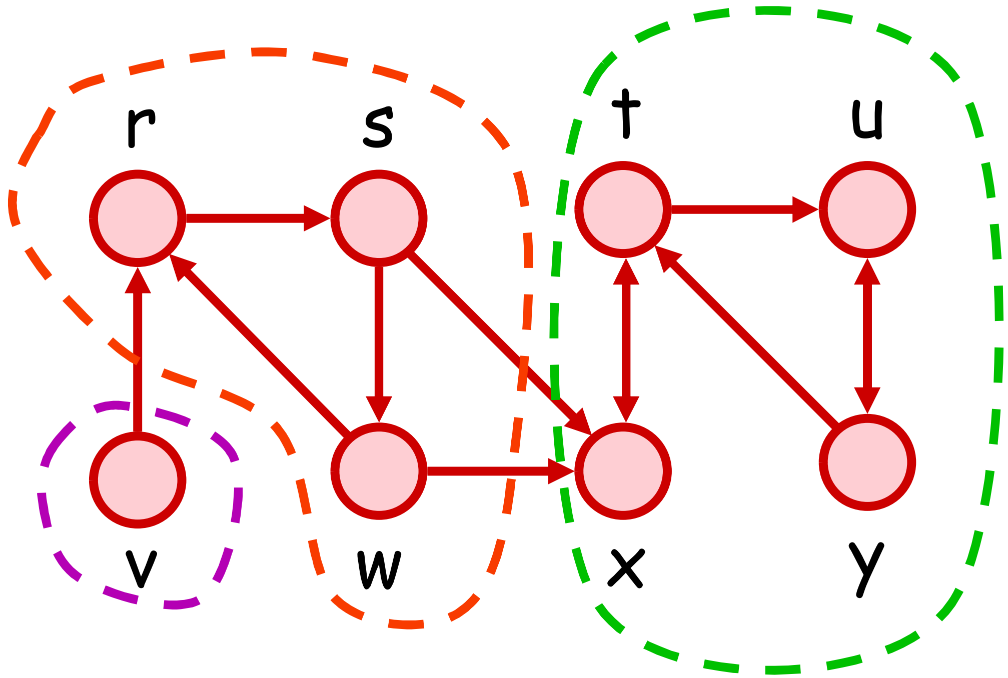
Mutually Reachable

- Let G be a directed graph
- Let u and v be two vertices in G
- Definition: If u can reach v (by a path) and v can reach u (by a path), then we say u and v are mutually reachable
- We shall use the notation $u \leftrightarrow v$ to indicate u and v are mutually reachable
- Also, we assume $u \leftrightarrow u$ for any node u

Strongly Connected Components

- Let $V_1, V_2, \ldots, V_k$ denote the partitions of a graph G
- Each V_i is called a strongly connected component (SCC) of G (i.e. vertices in V_i are mutually reachable)

e.g.,



Property of SCC

- Let $G = (V, E)$ be a directed graph
- Let G^T be a graph obtained from G by reversing the direction of every edge in G

  → Adjacency matrix of G^T

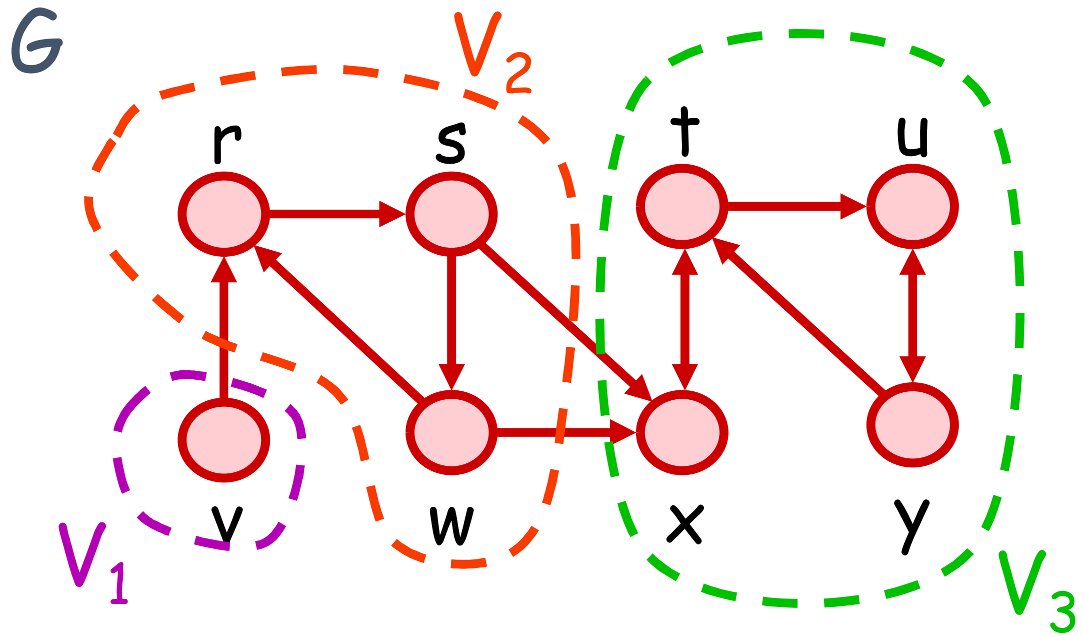
  = transpose of adjacency matrix of G

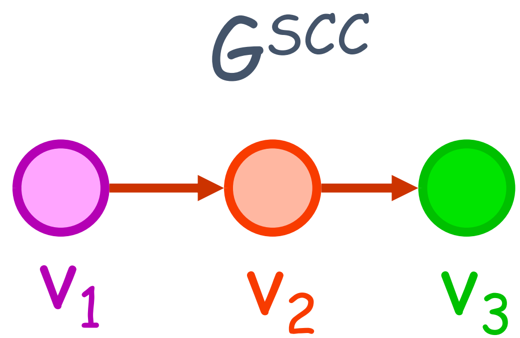
- Theorem:

  G and G^T has the same set of SCC's

Property of SCC

- Let $V_1, V_2, \ldots, V_k$ denote SCC of a graph G
- Let G^{SCC} be a simple graph obtained by contracting each V_i into a single vertex v_i
  - We call G^{SCC} the component graph of G





Property of G^{SCC}

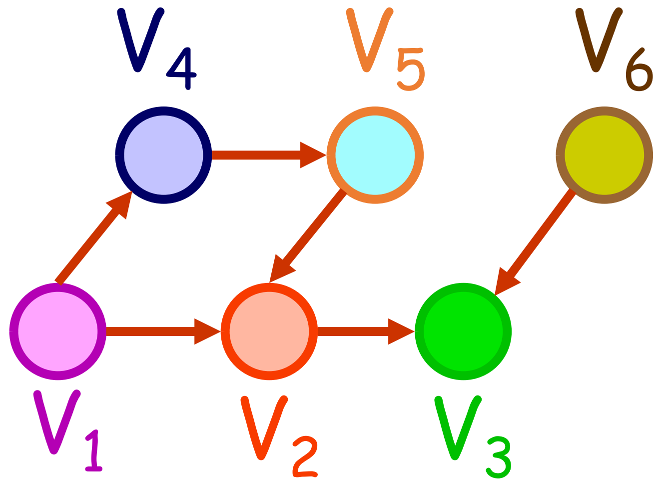
- Theorem: G^{SCC} is acyclic
- Proof: (By contradiction)

  If G^{SCC} has a cycle, then there are some vertices v_i and v_j with $v_i \leftrightarrow v_j$

  By definition, v_i and v_j correspond to two distinct SCC V_i and V_j in G. However, we see that any pair of vertices in V_i and V_j are mutually reachable → contradiction

Property of G^{SCC}

- Suppose the DAG (directed acyclic graph) on the right side is the G^{SCC} of some graph G



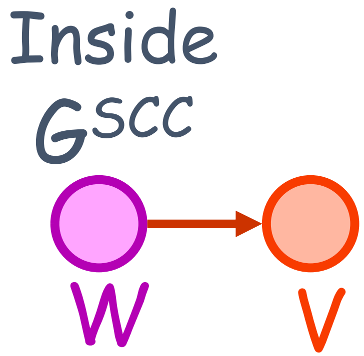
- Now, suppose we perform DFS on G
  - let u = node with largest finishing time
- Question: Which SCC can u be located ? (see next Lemma)

Property of G^{SCC}

- Lemma: Consider any graph G. Let G^{SCC} be its component graph. Suppose V is a vertex in G^{SCC} with at least one incoming edge. Then, the node u finishing last in any DFS of G cannot be a vertex of the SCC corresponding to V

Proof

- Let v = SCC corresponding to V
- Since V has incoming edge, there exists W such that (W, V) is an edge in G^{SCC}



- In the next two slides, we shall show that some node in SCC(W) must finish later than any node in SCC(V)
  - Consequently, u cannot be in SCC(V)

Proof

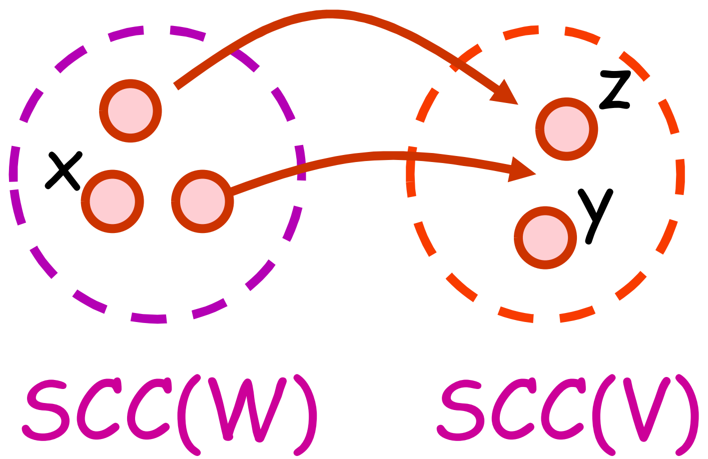
Let x = 1st node in SCC(W)
discovered by DFS

Let y = 1st node in SCC(V)
discovered by DFS

Let z = last node in SCC(V)
discovered by DFS

// Note: z may be the same as y

Inside G



- By white-path theorem, we must have

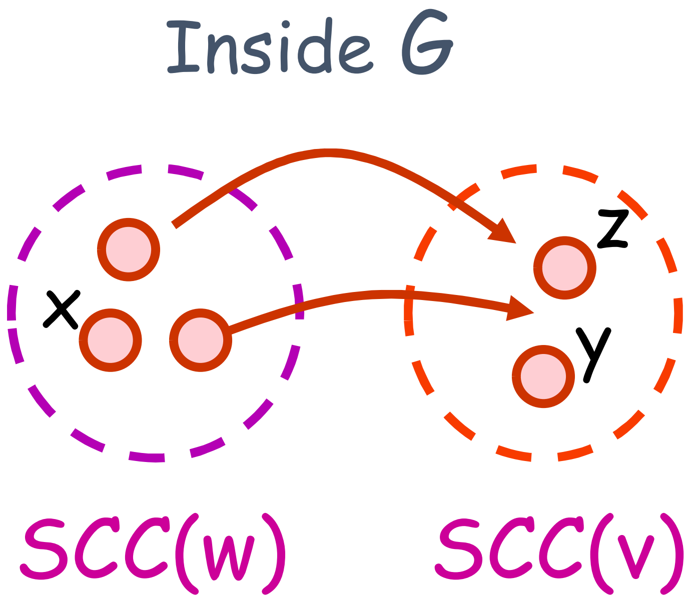
$$d(y) \leq d(z) < f(z) \leq f(y)$$

Proof

If $d(x) < d(y)$

- then y becomes x's descendant (by white-path)

  ➔ $f(z) \leq f(y) < f(x)$



If $d(y) < d(x)$

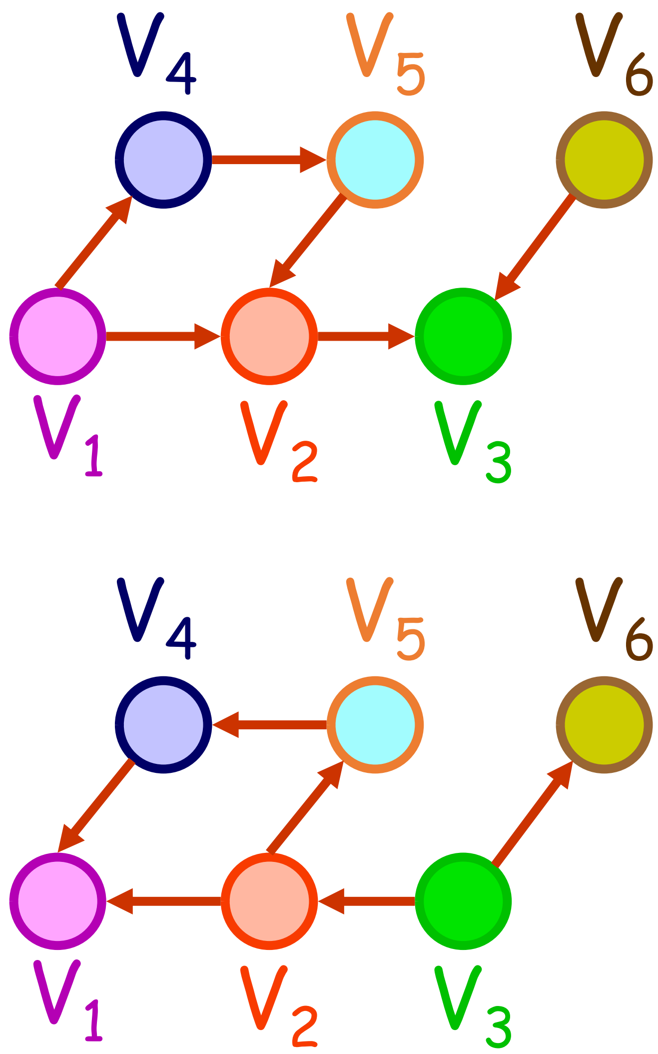
- since x cannot be y's descendant (otherwise, they are in the same SCC)

  ➔ $d(y) < f(y) < d(x) < f(x)$

  ➔ $f(z) \leq f(y) < f(x)$

Finding SCC

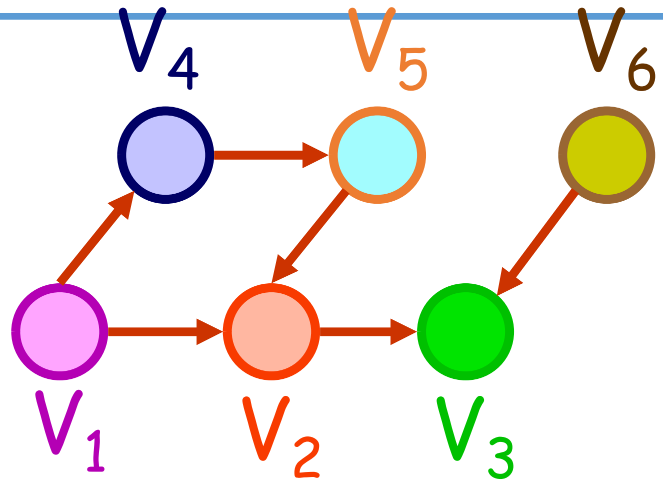
- So, we know that u (last finished node of G) must be in an SCC with no incoming edges
- Let us reverse edge directions, and start DFS on G^T from u
- Question: Who will be u's descendants ??



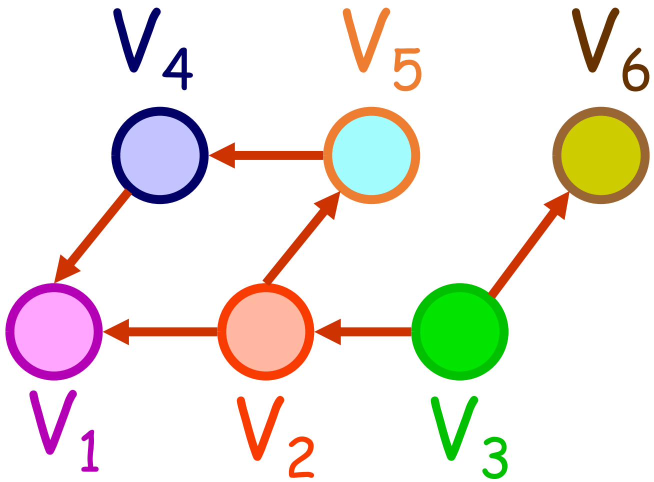
New G^{SCC}

Finding SCC

- Note that nodes in the SCC containing u cannot connect to nodes in other SCCs in G^T



- By white-path theorem, the descendants of u in G^T must be exactly those nodes in the same SCC as u



New G^{SCC}

Finding SCC

- Once DFS on u inside G^T has finished, all nodes in the same SCC as u are finished

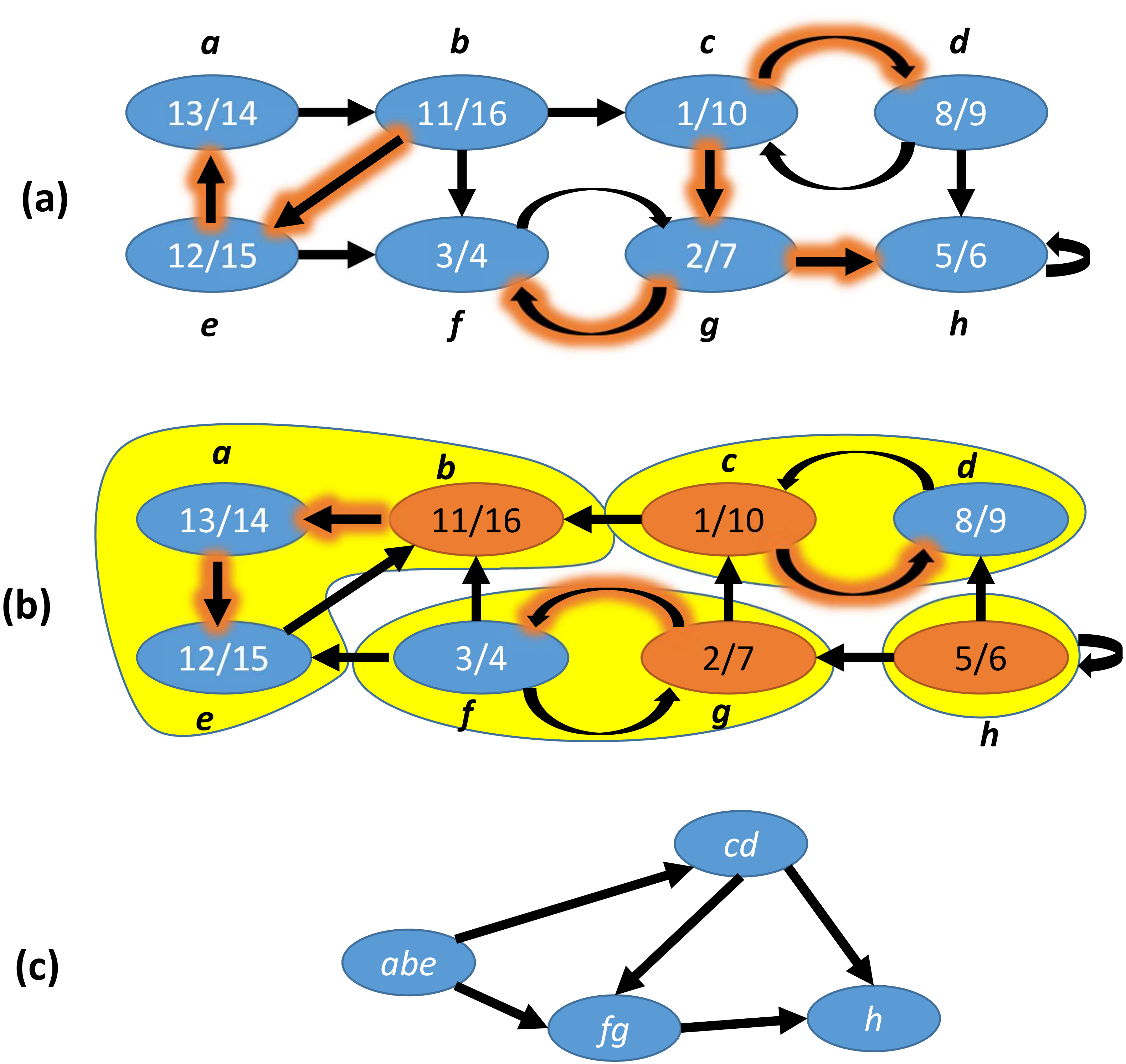
  → Any subsequent DFS in G^T will be made as if this SCC was removed from G^T

- Now, let u' be the remaining node in G^T whose finishing time (in DFS in G) is latest
  - Where can u' be located?
  - Who will be the descendents of u' if we perform DFS in G^T now?

- Our observations lead to the following algorithm for finding all SCCs of G :

Finding-all-SCC(G) {

1. Call DFS(G) to compute finish times u.f for each vertex u;
2. Construct G^T ;
3. Call (G^T) from u.f in decreasing order ;
4. Output the vertices of each tree in the depth-first forest formed in line 3 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
(b)
a
b
c
d
13/14
11/16
1/10
8/9
12/15
3/4
2/7
5/6
e
f
g
h
(c)
cd
abe
fg
h

Correctness & Performance

- The correctness of the algorithm can be proven by induction

  (Hint: Show that at each sub-search in Step 3, u is chosen from an SCC which has no outgoing edges to any nodes in an "unvisited" SCC of G^T.

  ➔ By white-path theorem, exactly all nodes in the same SCC become u's descendants)

- Running Time: $O(|V|+|E|)$ (why?)

Practice at home

- Exercises: 22.5-1, 22.5-3