CS473-Algorithms I

Lecture 16

Strongly Connected Components

Strongly Connected Components

Definition: a strongly connected component (SCC) of a directed graph $G=(V,E)$ is a maximal set of vertices $U \subseteq V$ such that

- For each $u,v \in U$ we have both $u \mapsto v$ and $v \mapsto u$

  i.e., u and v are mutually reachable from each other ($u \leftrightarrows v$)

Let $G^T=(V,E^T)$ be the *transpose* of $G=(V,E)$ where

$$E^T = \{(u,v) : (v,u) \in E\}$$

- i.e., E^T consists of edges of G with their directions reversed

  Constructing G^T from G takes $O(V+E)$ time (adjacency list rep)

  Note: G and G^T have the same SCCs ($u \leftrightarrows v$ in $G \Leftrightarrow u \leftrightarrows v$ in G^T)

Strongly Connected Components

Algorithm

(1) Run **DFS**(G) to compute finishing times for all $u \in V$

(2) Compute G^T

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing $f[u]$ computed in Step (1)

(4) Output vertices of each DFT in DFF of Step (3) as a separate SCC

Strongly Connected Components

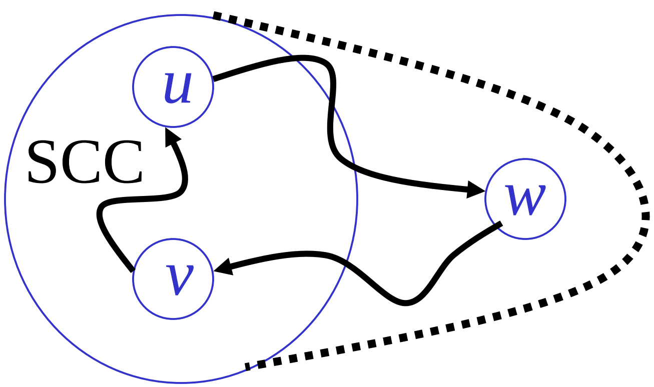
Lemma 1: no path between a pair of vertices in the same SCC, ever leaves the SCC

Proof: let u and v be in the same SCC $\Rightarrow u \leftrightarrows v$

let w be on some path $u \mapsto w \mapsto v \Rightarrow u \mapsto w$

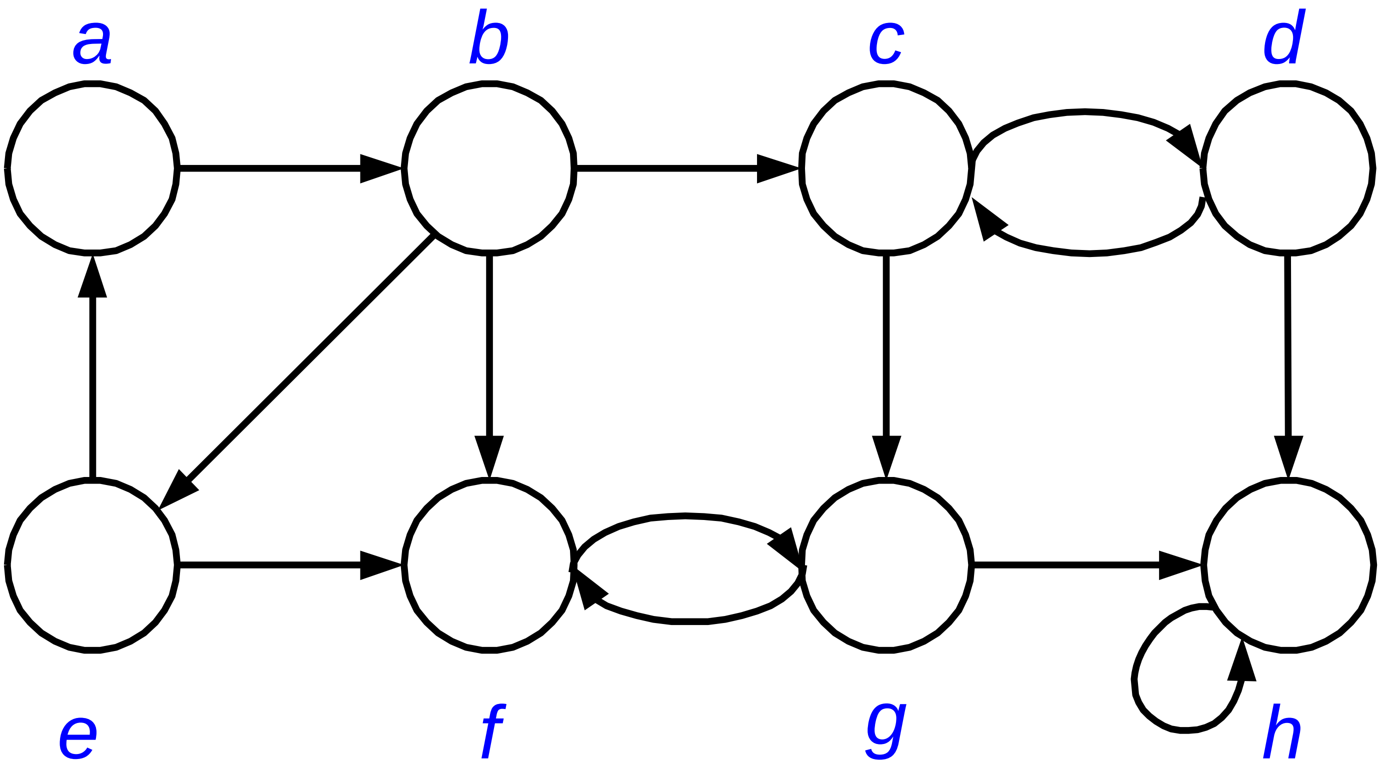
but $v \mapsto u \Rightarrow \exists$ a path $w \mapsto v \mapsto u \Rightarrow w \mapsto u$

therefore u and w are in the same SCC



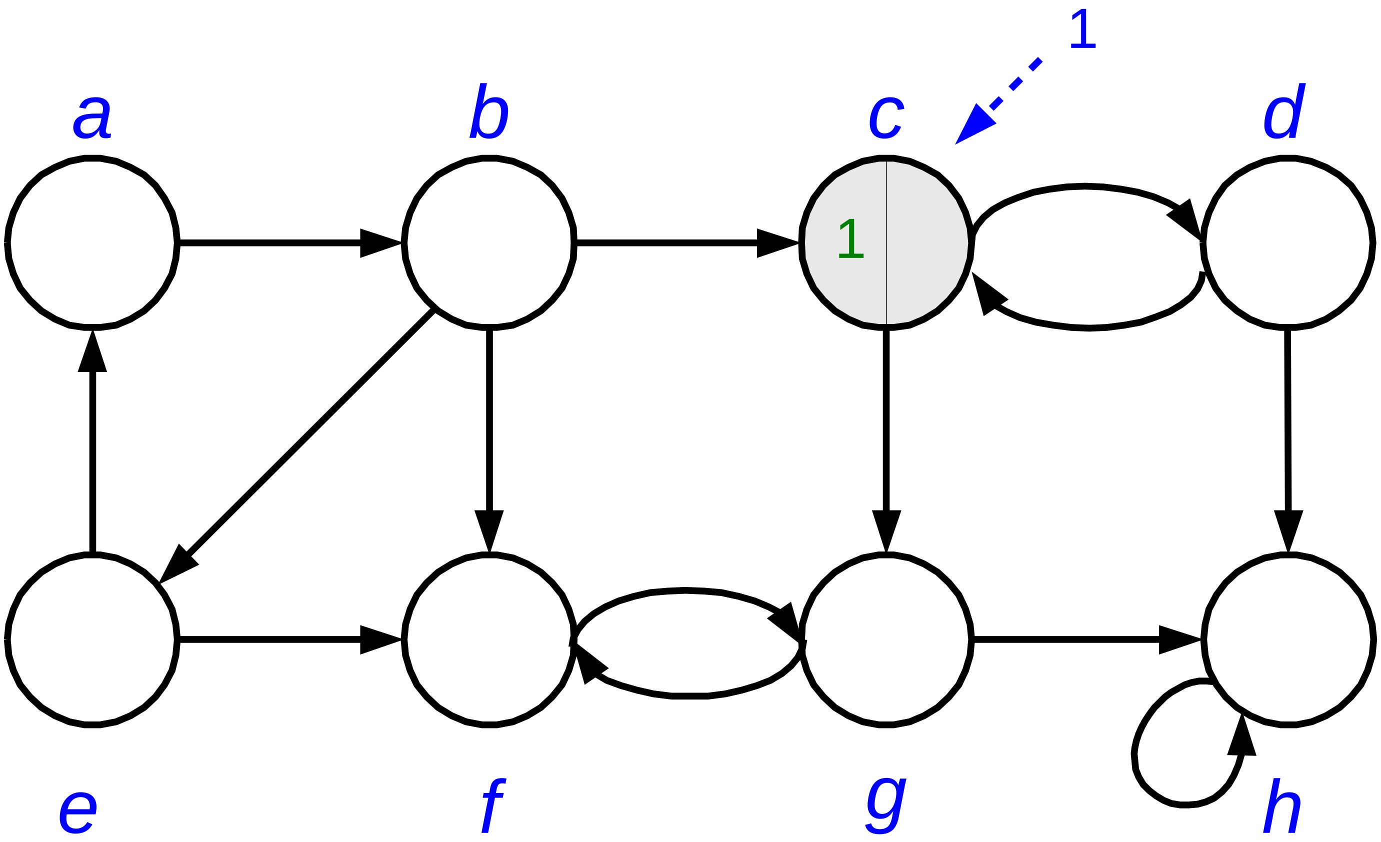
QED

SCC: Example



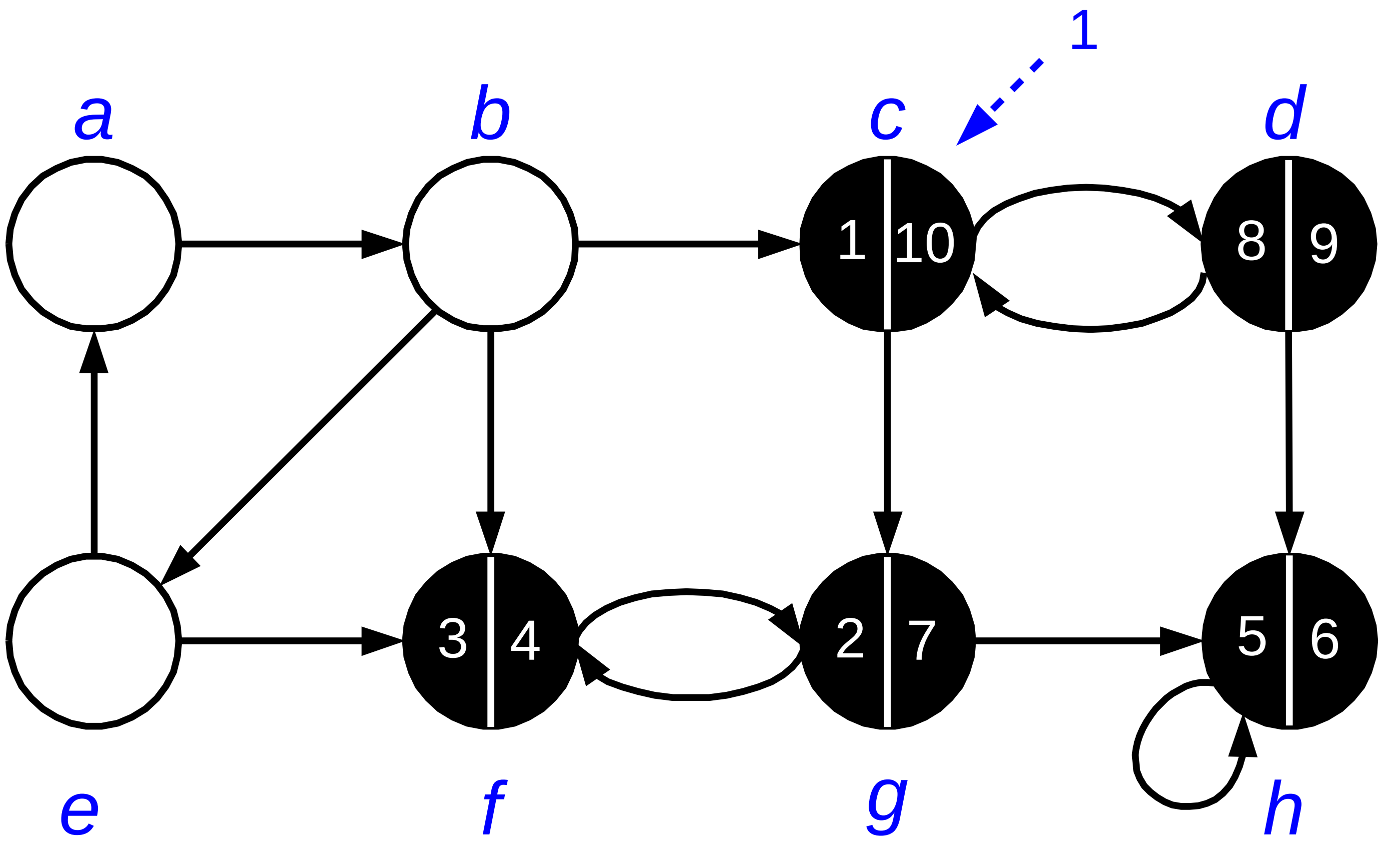
SCC: Example

(1) Run **DFS**(G) to compute finishing times for all $u \in V$



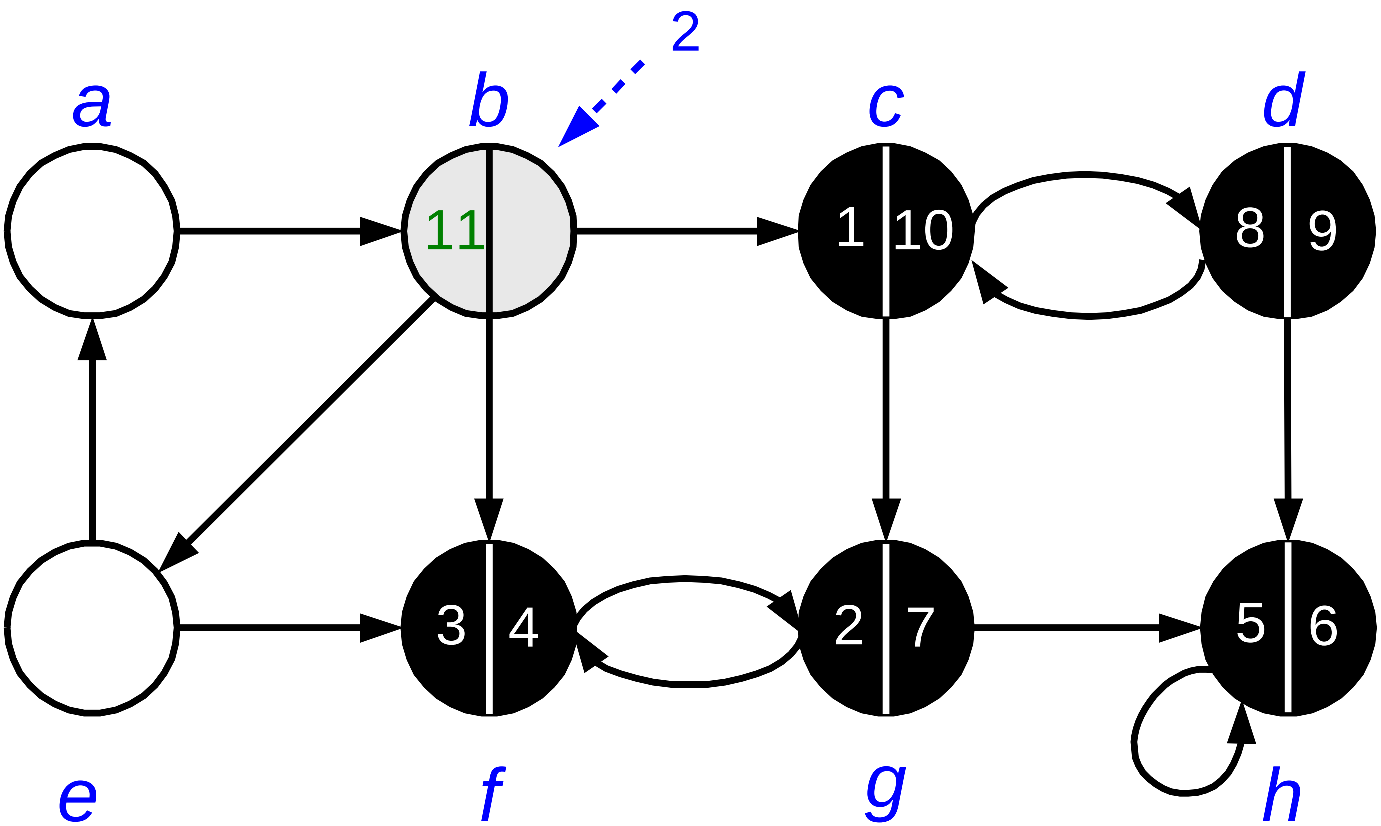
SCC: Example

(1) Run **DFS**(G) to compute finishing times for all $u \in V$

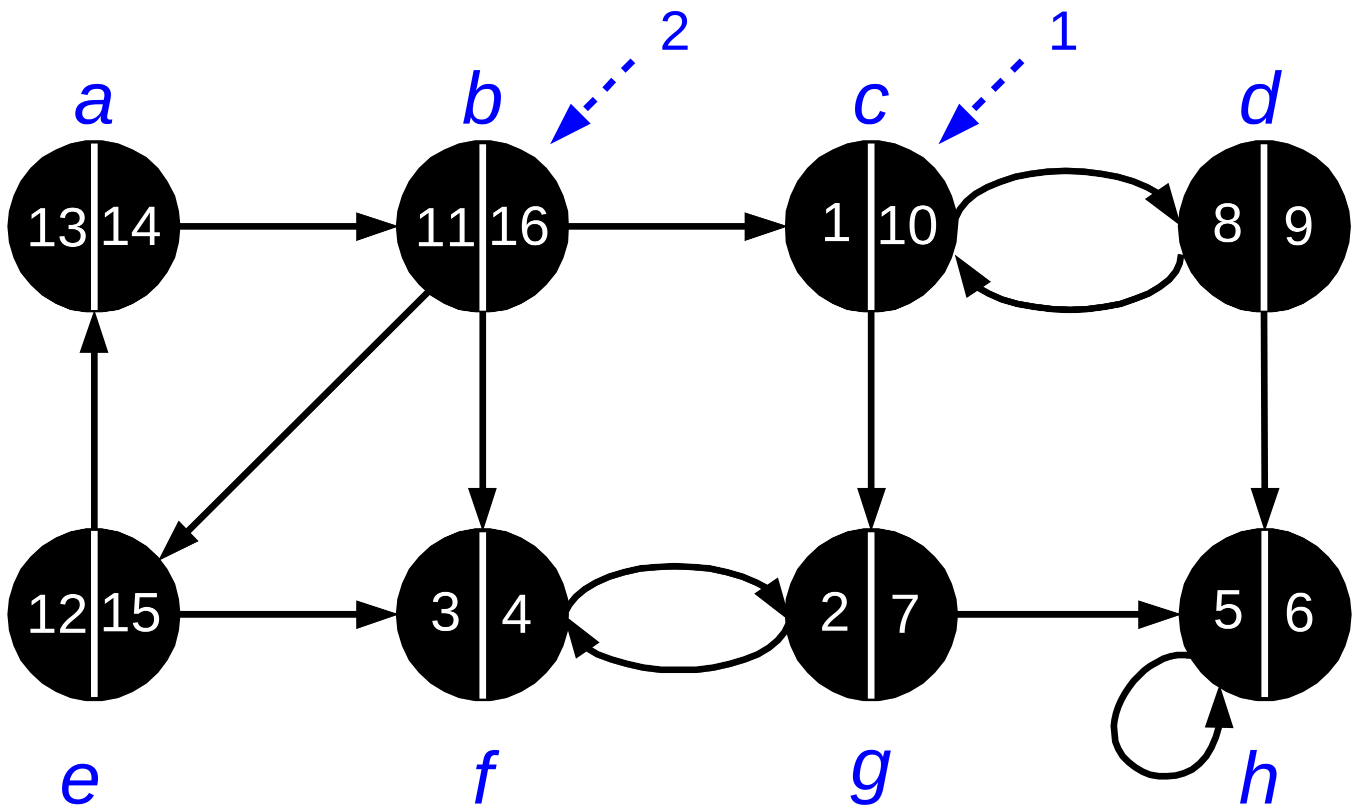


SCC: Example

(1) Run **DFS**(G) to compute finishing times for all $u \in V$



SCC: Example

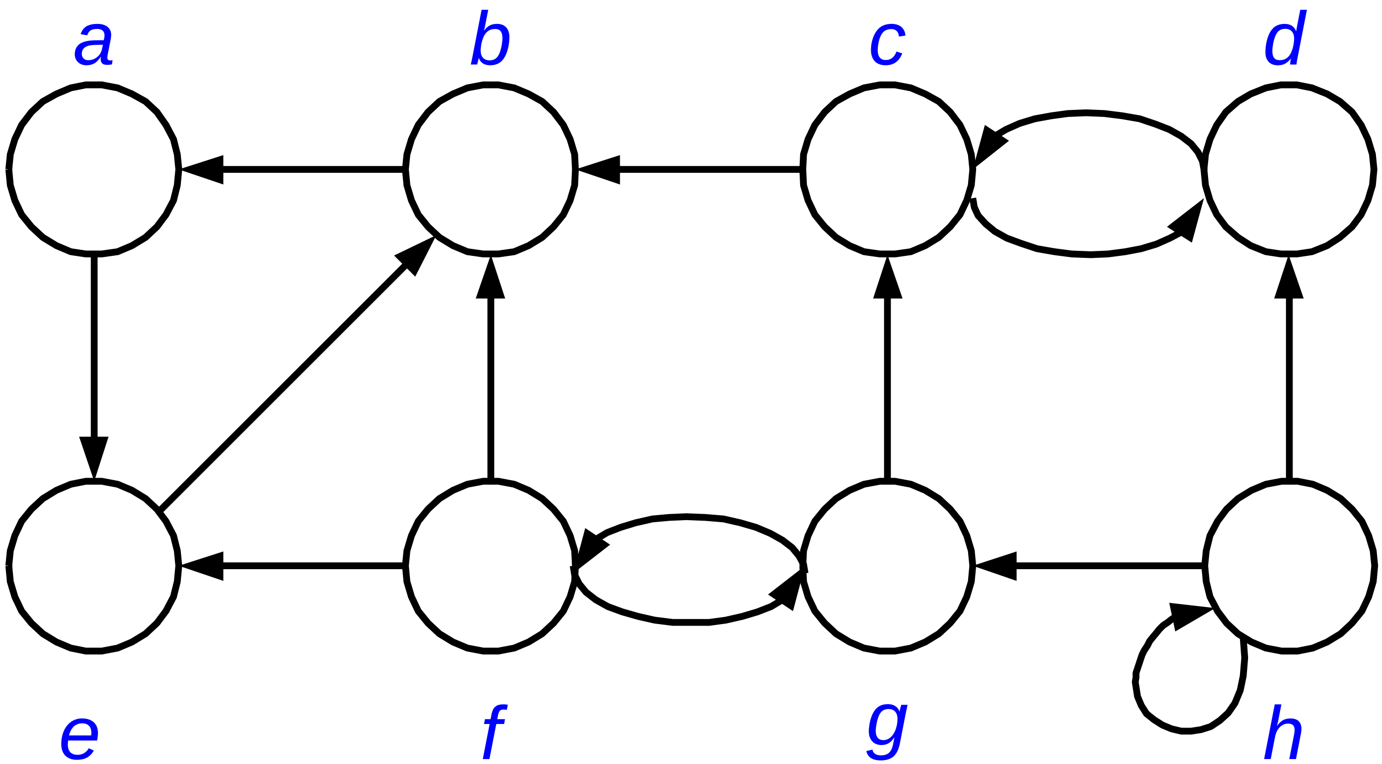


Vertices sorted according to the finishing times:

$\langle \boldsymbol{b}, \boldsymbol{e}, \boldsymbol{a}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{g}, \boldsymbol{h}, \boldsymbol{f} \rangle$

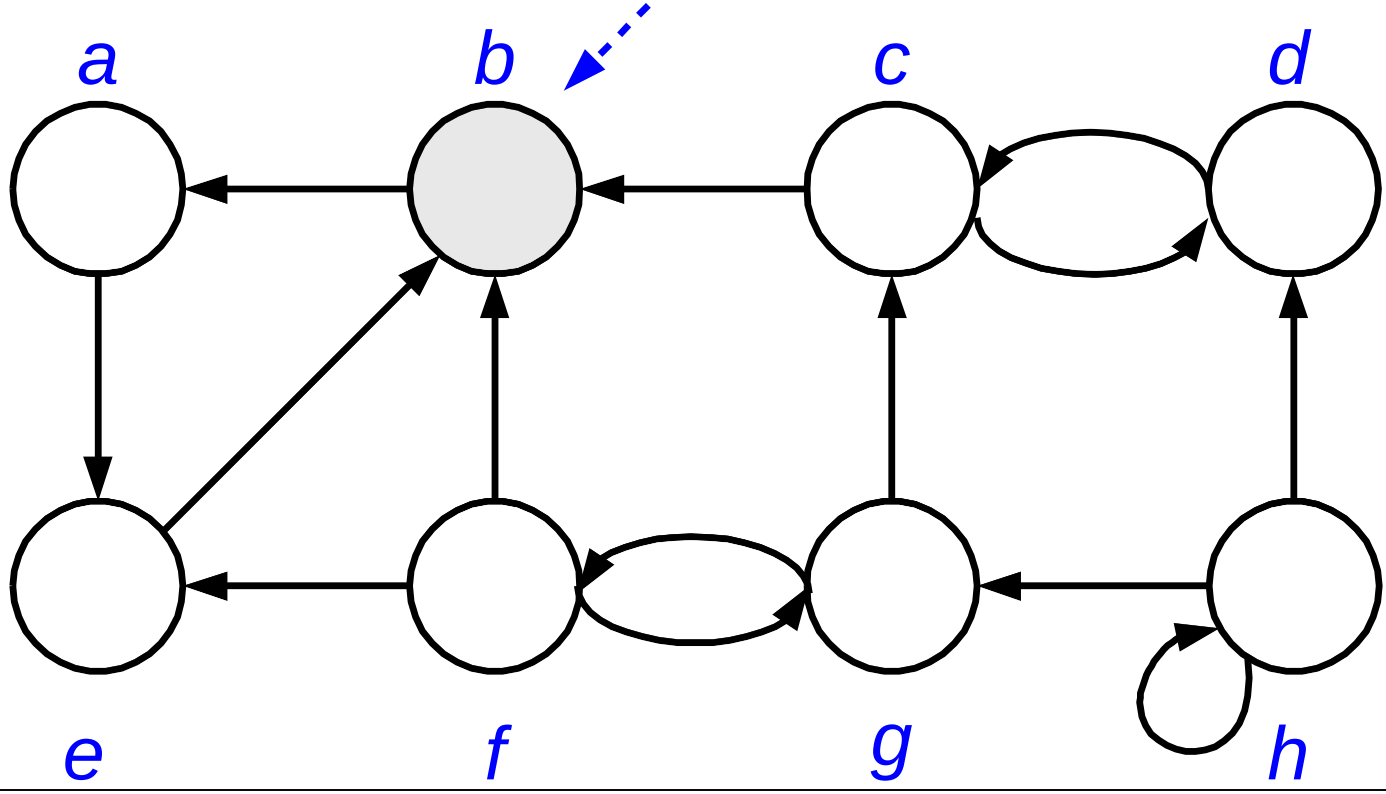
SCC: Example

(2) Compute G^T



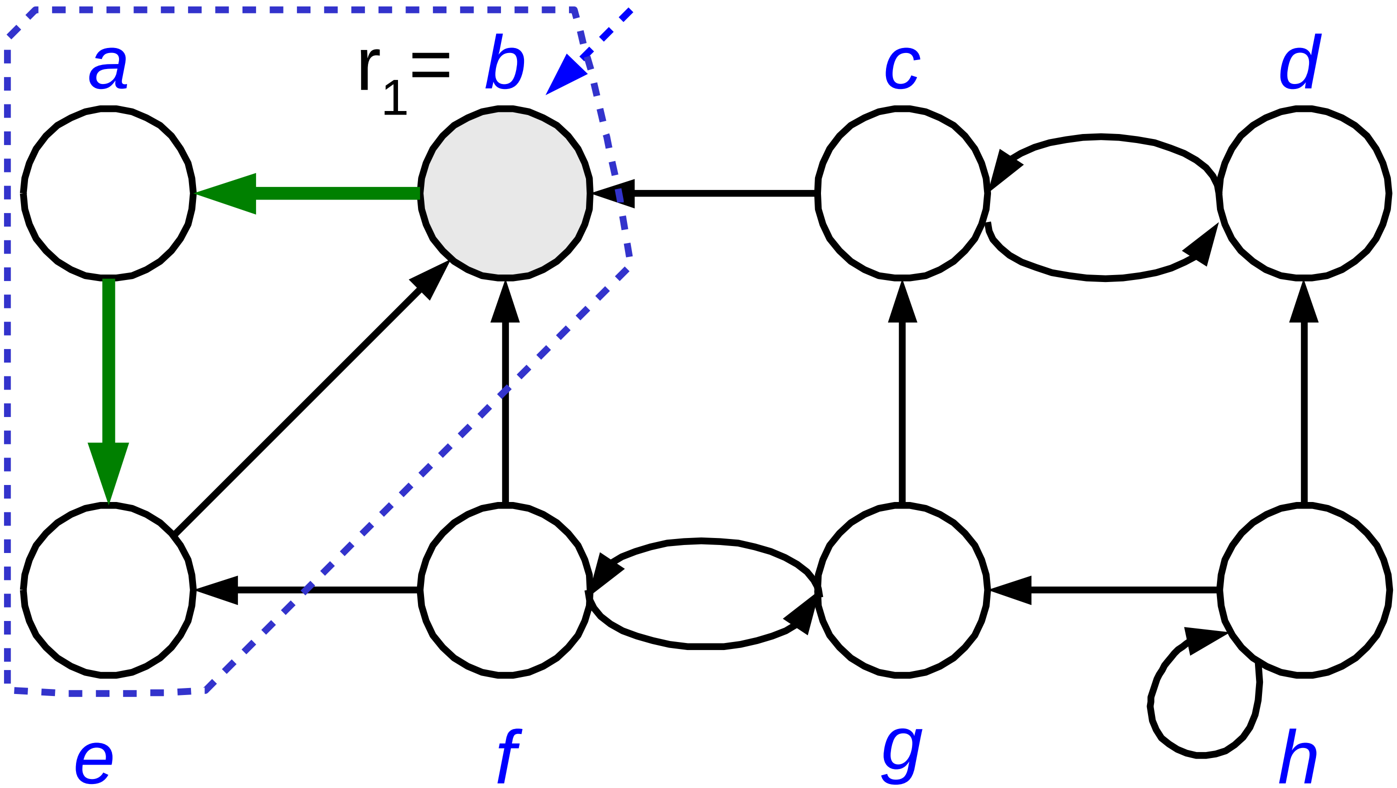
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[u] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



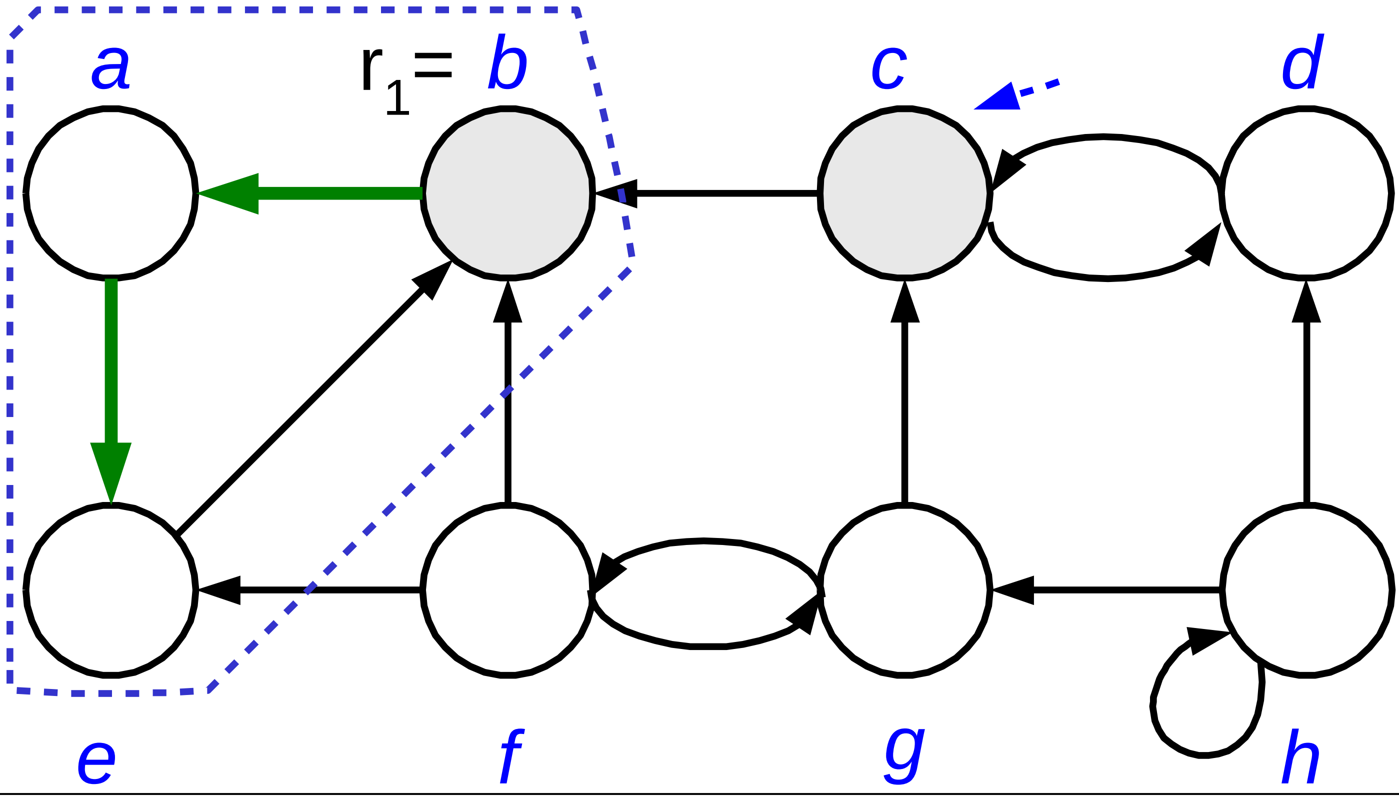
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[*u*] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



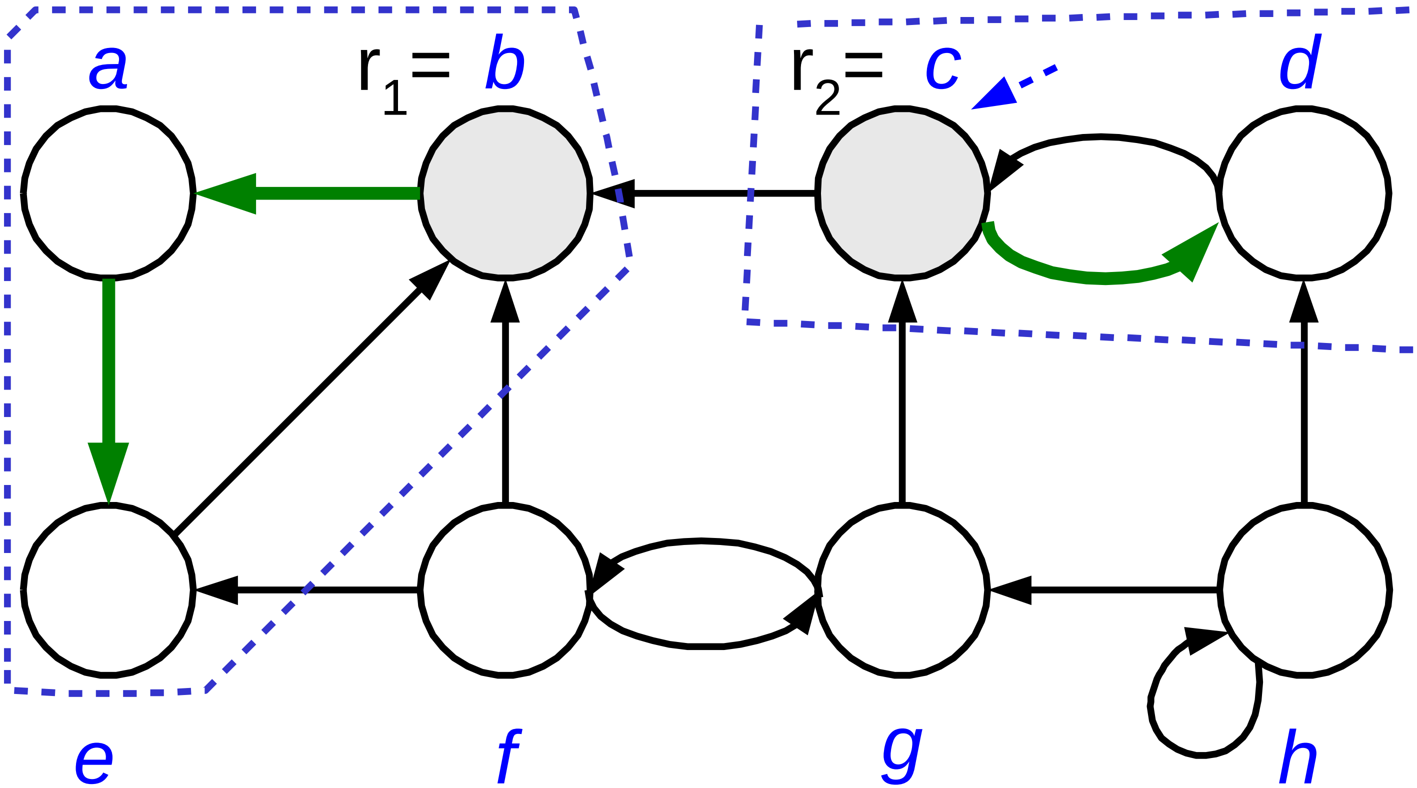
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[*u*] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



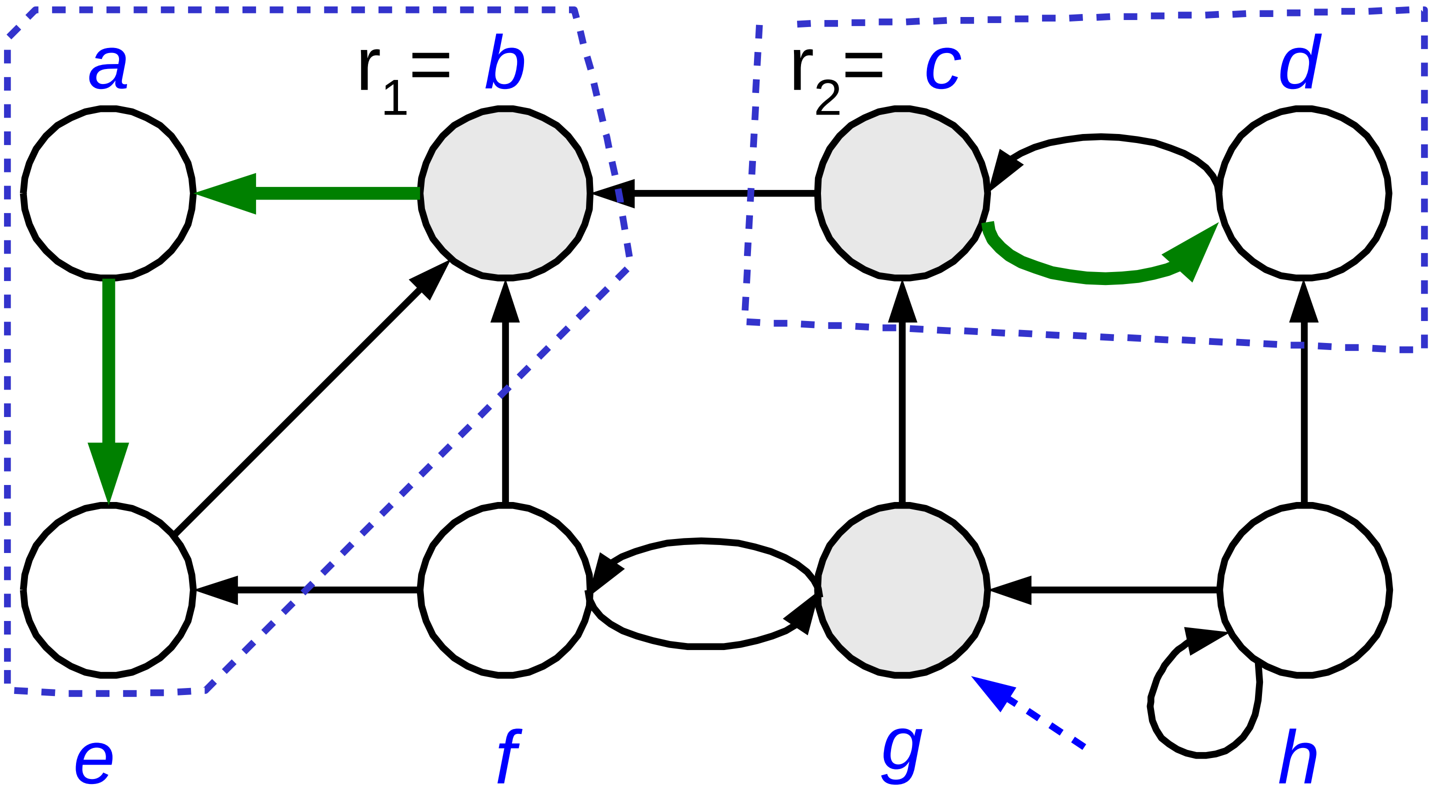
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[*u*] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



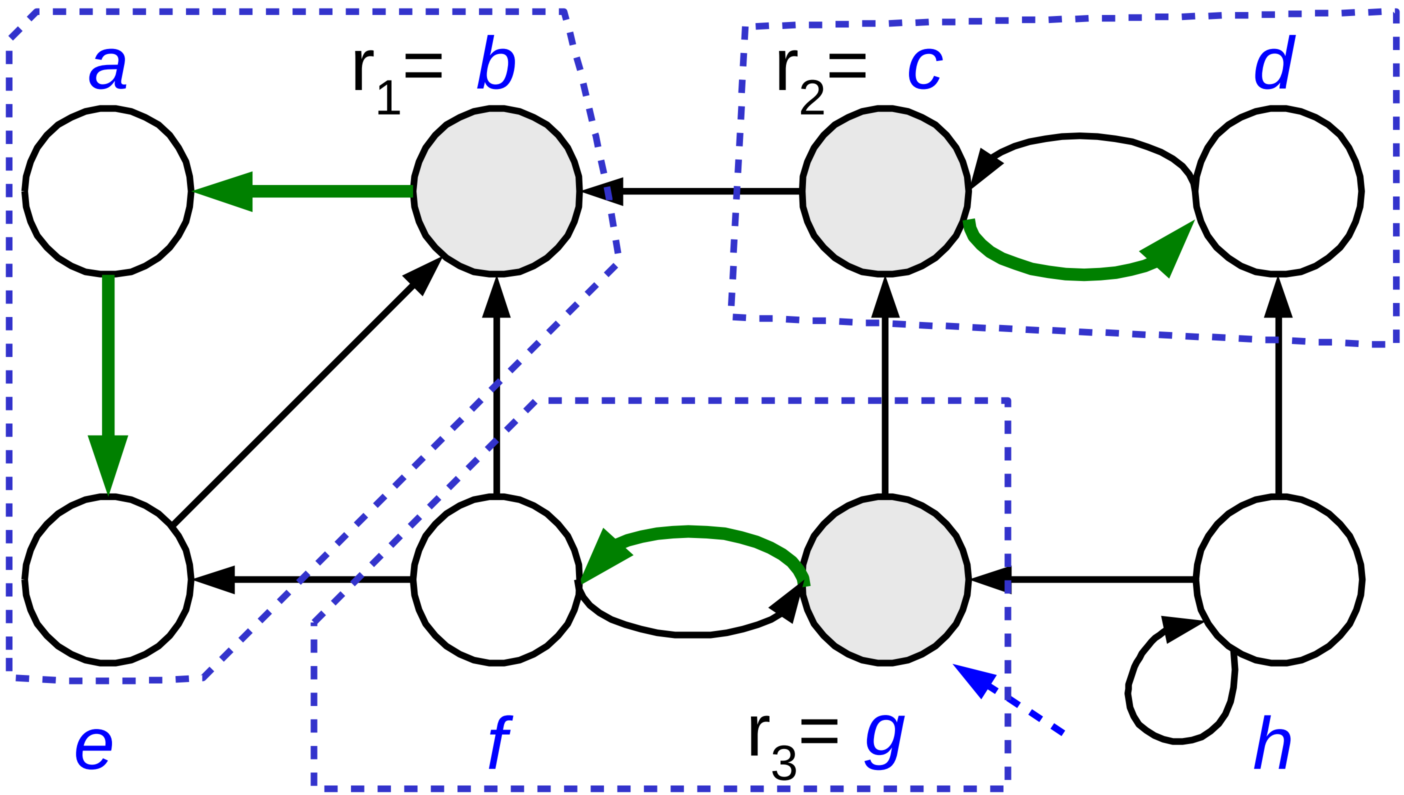
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[*u*] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



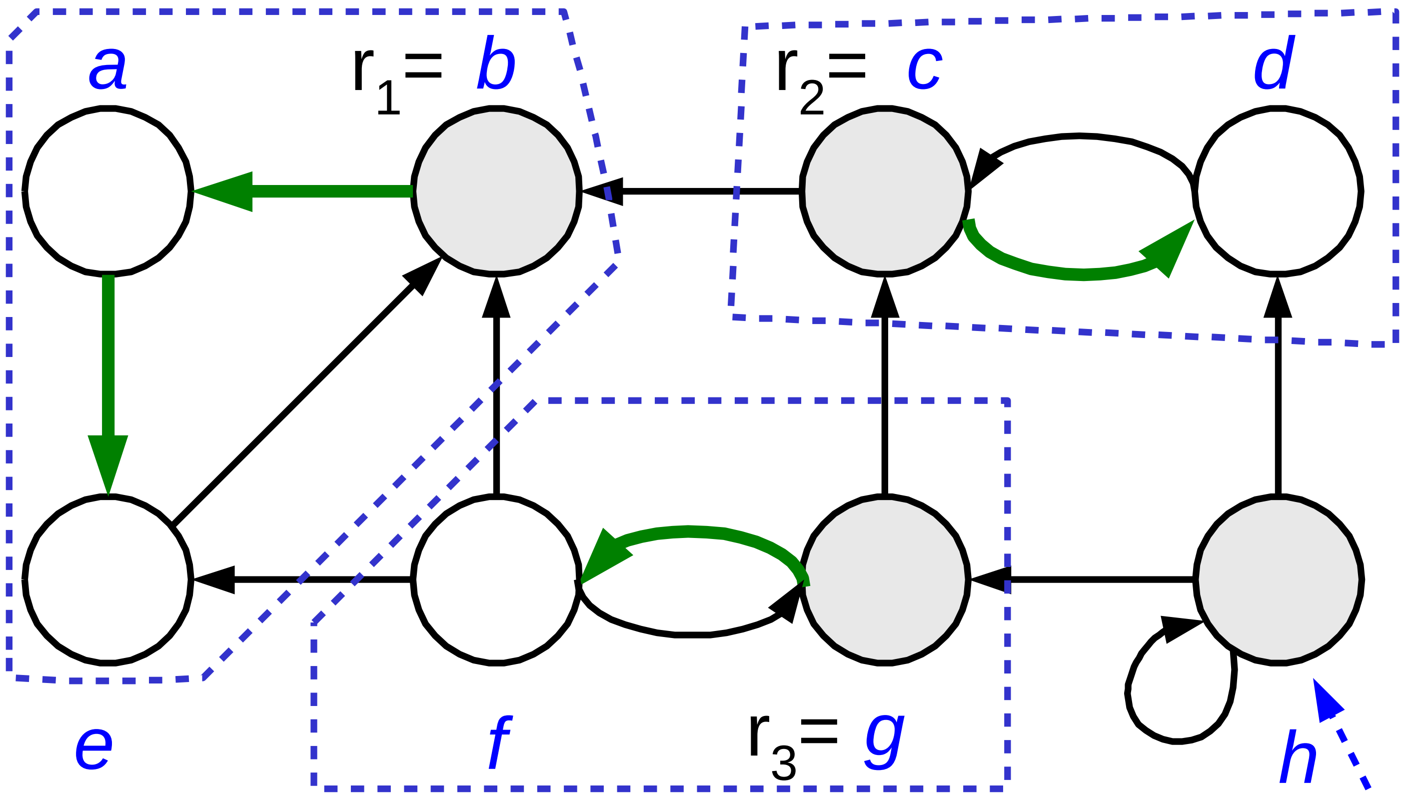
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[*u*] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



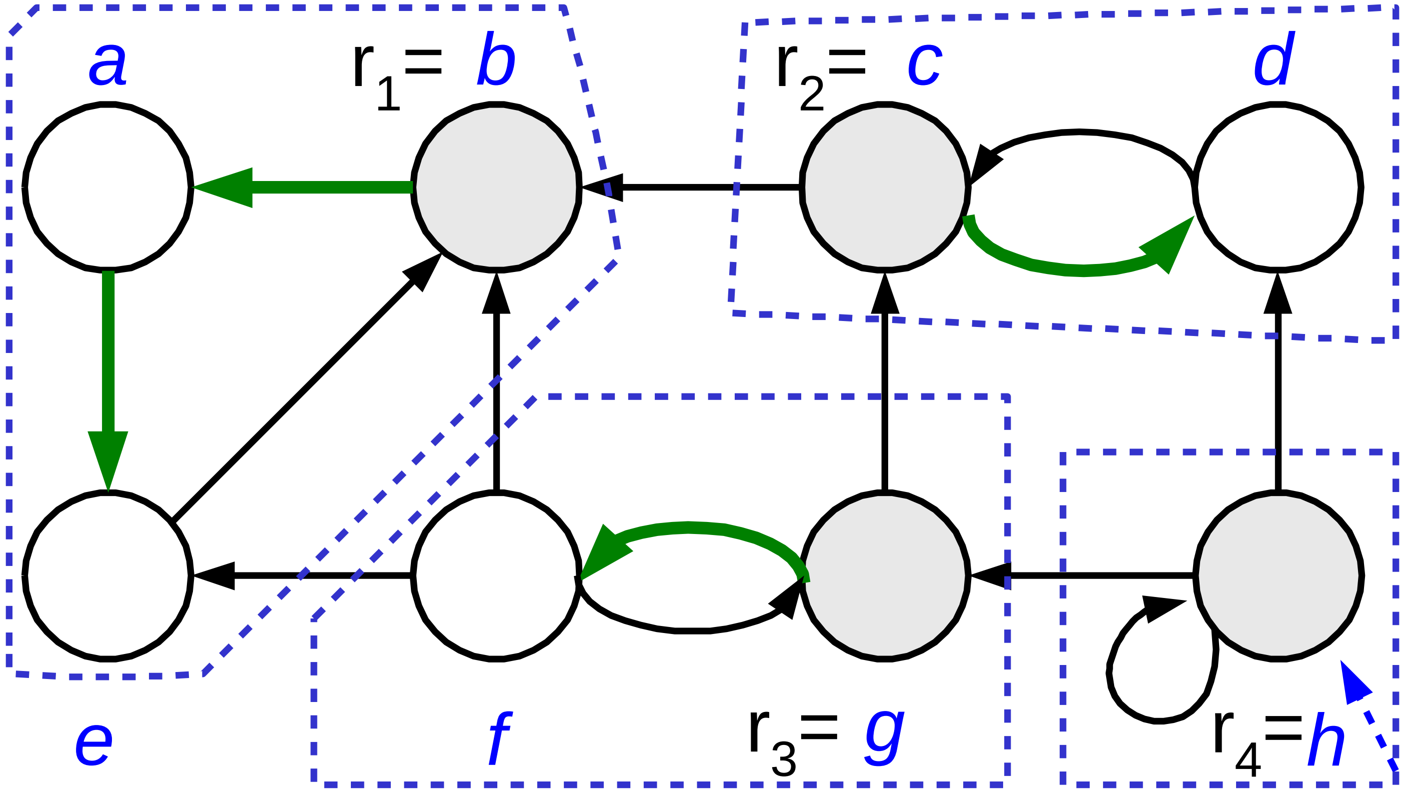
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[u] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩



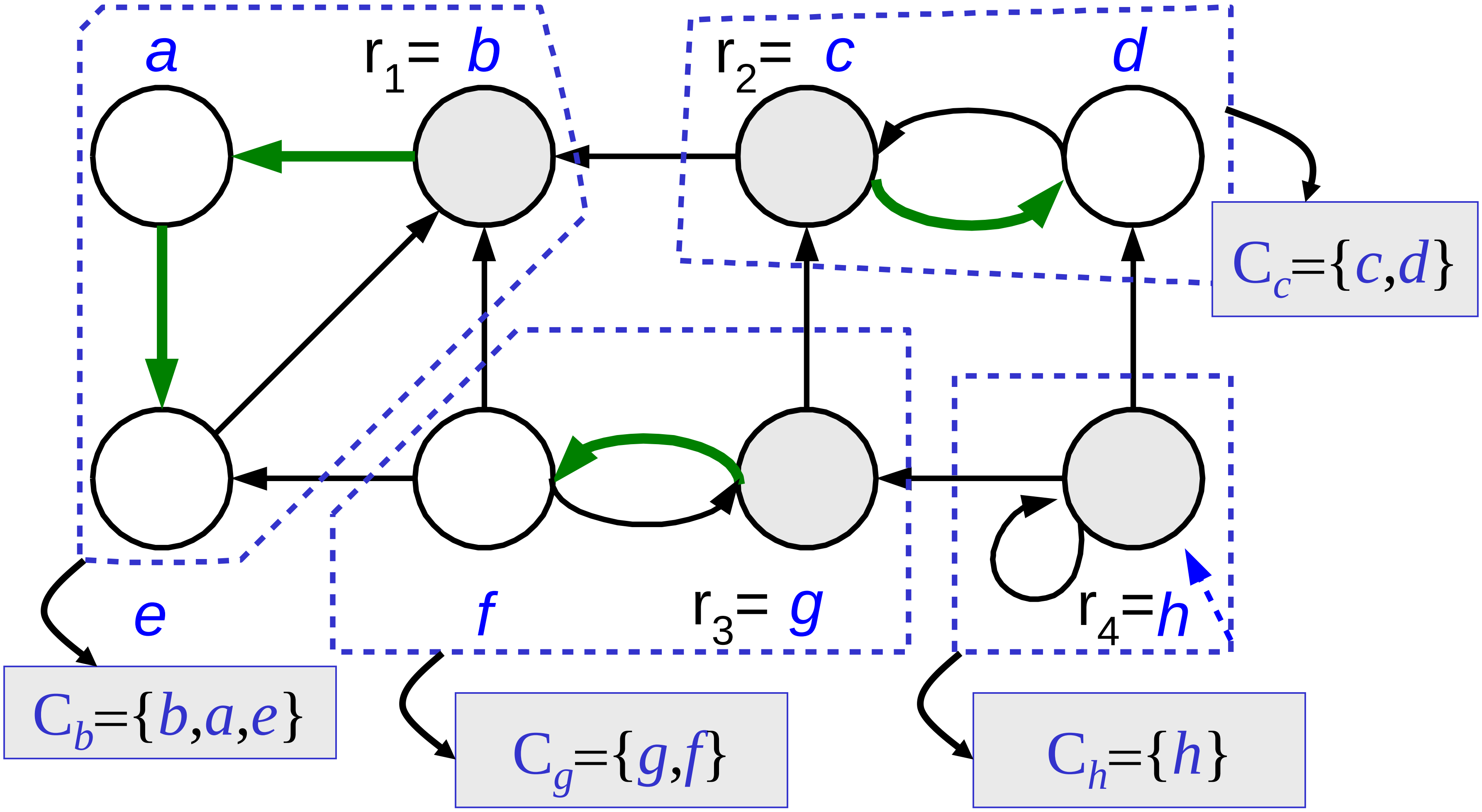
SCC: Example

(3) Call **DFS**(G^T) processing vertices in main loop in decreasing f[u] order: ⟨***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** ⟩

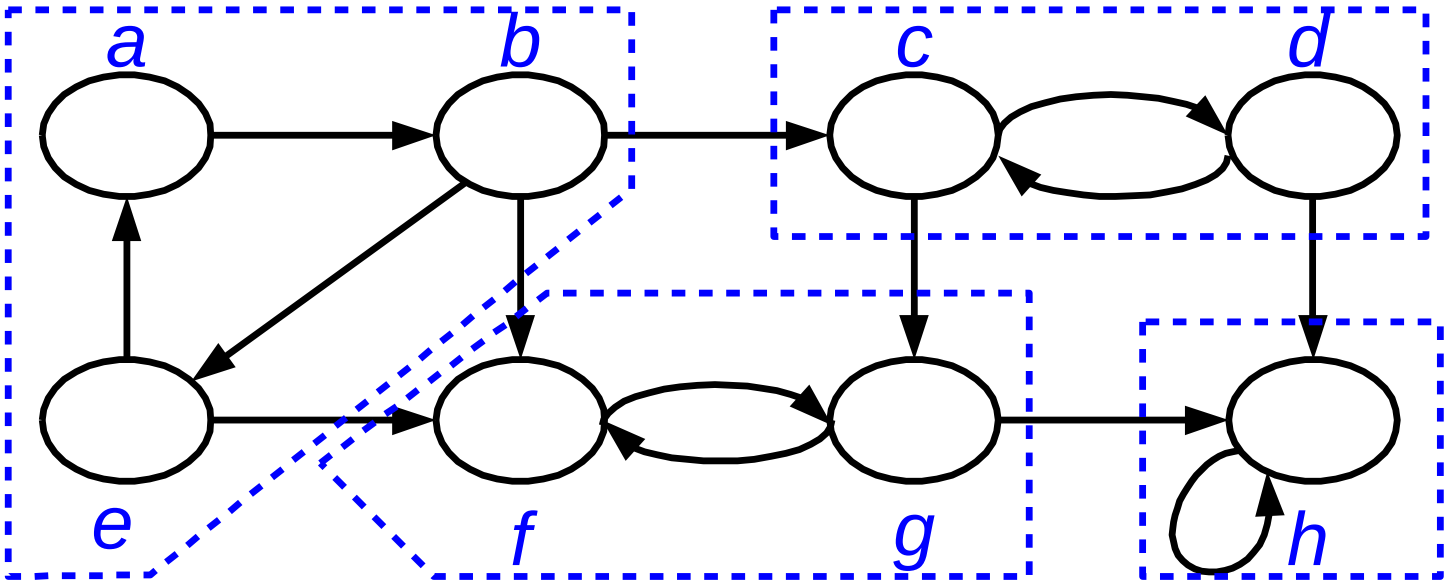


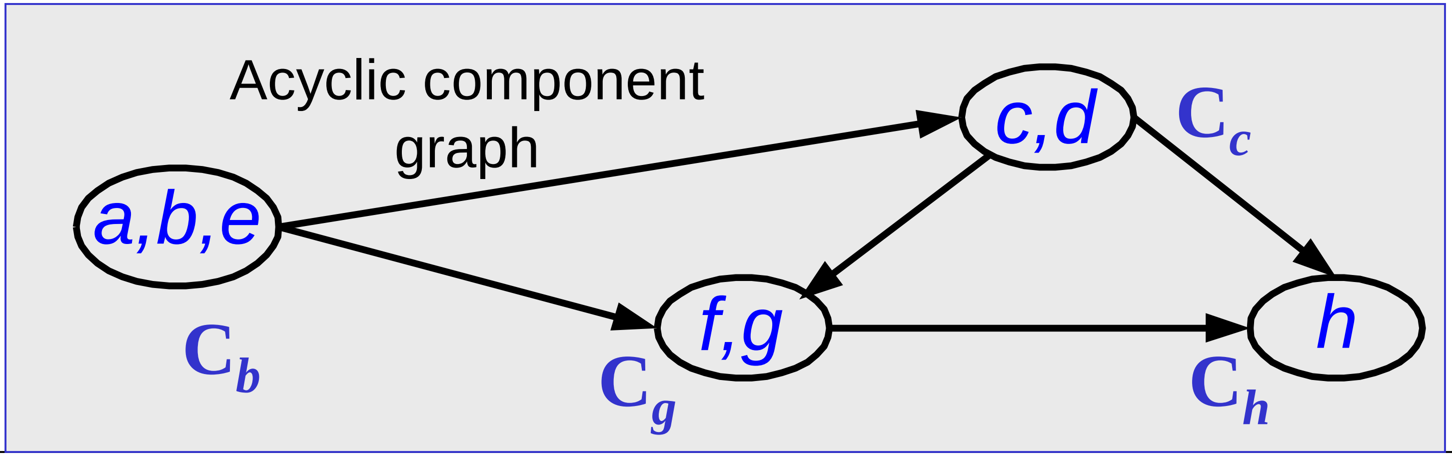
SCC: Example

(4) Output vertices of each DFT in DFF as a separate SCC



SCC: Example





Strongly Connected Components

Thrm 1: in any DFS, all vertices in the same SCC are placed in the same DFT

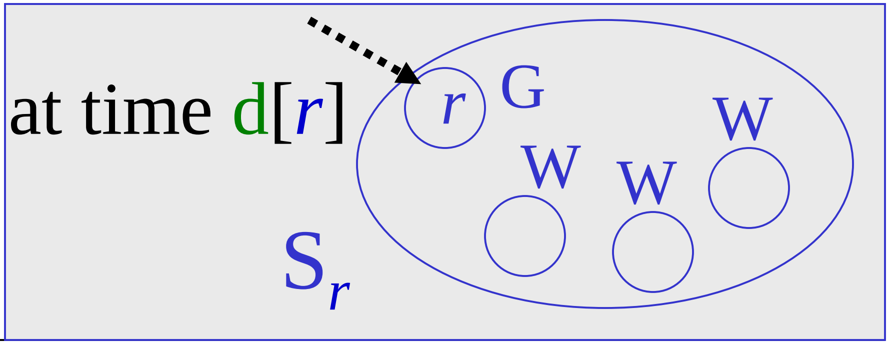
Proof: let r be the first vertex discovered in SCC S_r

because r is first, color[x]=WHITE $\forall x \in S_r - \{r\}$ at time d[r]

So all vertices are WHITE on each $r \mapsto x$ path $\forall x \in S_r - \{r\}$

– since these paths never leave S_r

Hence each vertex in $S_r - \{r\}$ becomes a descendent of r (White-path Thrm)



QED

Notation for the Rest of This Lecture

- d[u] and f[u] refer to those values computed by **DFS**(G) at step (1)
- $u \mapsto v$ refers to G not G^T

Definition: forefather $\phi(u)$ of vertex u

1. $\phi(u)$ = That vertex w such that $u \mapsto w$ and f[u] is maximized
2. $\phi(u) = u$ possible because $u \mapsto u \Rightarrow$ $\boxed{f[u] \leq f[\phi(u)]}$

Strongly Connected Components

Lemma 2: $\phi(\phi(u)) = \phi(u)$

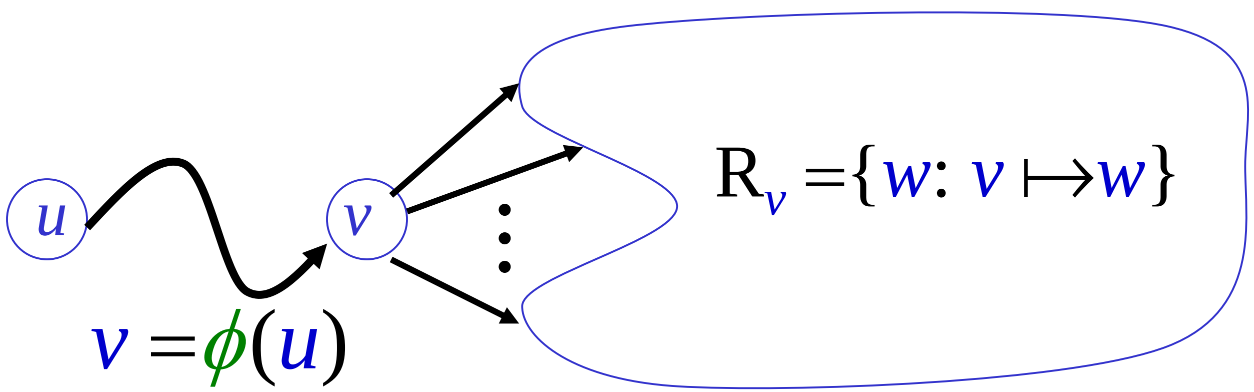
Proof try to show that $f[\phi(\phi(u))] = f[\phi(u)]$:

For any $u,v \in V$; $u \mapsto v \Rightarrow R_v \subseteq R_u \Rightarrow f[\phi(v)] \leq f[\phi(u)]$

So, $u \mapsto \phi(u) \Rightarrow f[\phi(\phi(u))] \leq f[\phi(u)]$

Due to definition of $\phi(u)$ we have $f[\phi(\phi(u))] \geq f[\phi(u)]$

Therefore $f[\phi(\phi(u))] = f[\phi(u)]$ QED



Note:
$f[x] = f[y] \Rightarrow$
$x = y$
(same vertex)

Strongly Connected Components

Properties of forefather:

- Every vertex in an SCC has the same forefather which is in the SCC
- Forefather of an SCC is the representative vertex of the SCC
- In the DFS of G, forefather of an SCC is the
  - first vertex discovered in the SCC
  - last vertex finished in the SCC

Strongly Connected Components

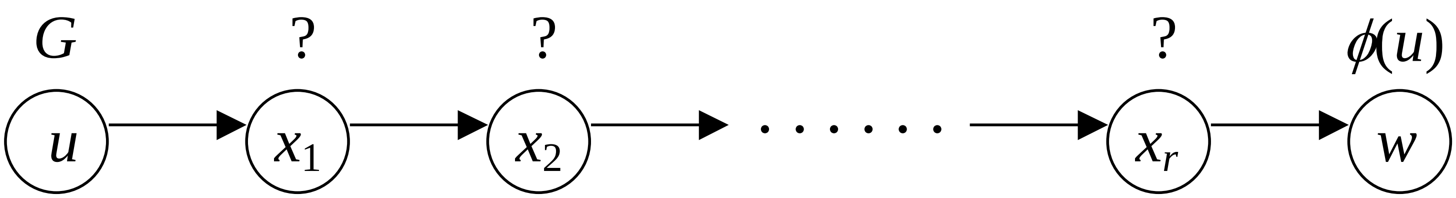
THM2: $\phi(u)$ of any $u \in V$ in any DFS of G is an ancestor of u

PROOF: Trivial if $\phi(u) = u$.

If $\phi(u) \neq u$, consider color of $\phi(u)$ at time d[u]

- $\phi(u)$ is GRAY: $\phi(u)$ is an ancestor of $u \Rightarrow$ proving the theorem
- $\phi(u)$ is BLACK: $f[\phi(u)] < f[u] \Rightarrow$ contradiction to def. of $\phi(u)$
- $\phi(u)$ is WHITE: $\exists$ 2 cases according to colors of intermediate vertices on p(u, $\phi(u)$)

Path p(u, $\phi(u)$) at time d[u]:



Strongly Connected Components

Case 1: every intermediate vertex $x_i \in$ p(u, $\phi(u)$) is WHITE

$\Rightarrow$ $\phi(u)$ becomes a descendant of u (WP-THM)

$\Rightarrow f[\phi(u)] < f[u]$

$\Rightarrow$ contradiction

Case 2: $\exists$ some non-WHITE intermediate vertices on p(u, $\phi(u)$)

- Let x_t be the last non-WHITE vertex on p(u, $\phi(u)$) $= \langle u, x_1, x_2, \ldots, x_r, \phi(u)\rangle$
- Then, x_t must be GRAY since BLACK-to-WHITE edge (x_t, x_{t+1}) cannot exist
- But then, p(x_t, $\phi(u)$) $= \langle x_{t+1}, x_{t+2}, \ldots, x_r, \phi(u)\rangle$ is a white path

$\Rightarrow$ $\phi(u)$ is a descendant of x_t (by white-path theorem)

$\Rightarrow f[x_t] > f[\phi(u)]$

$\Rightarrow$ contradicting our choice for $\phi(u)$ Q.E.D.

Strongly Connected Components

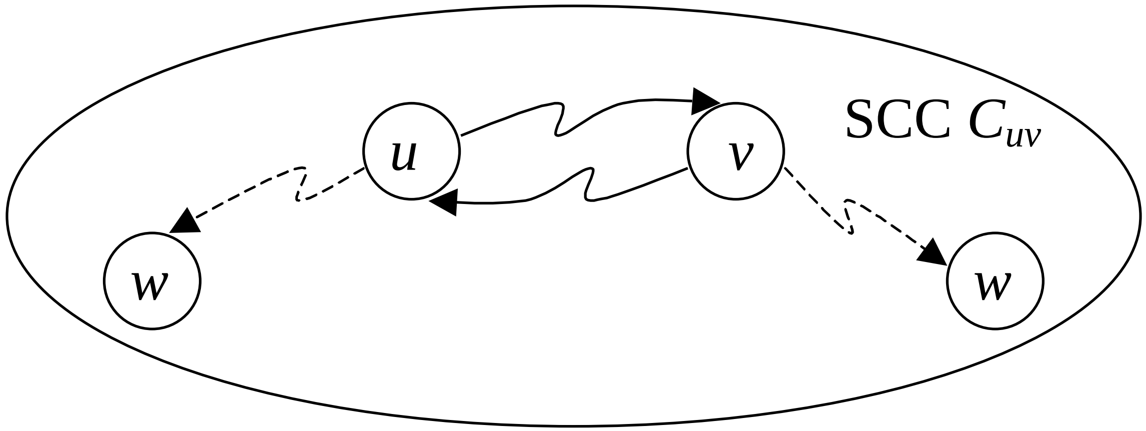
C1: in any DFS of $G = (V, E)$ vertices u and $\phi(u)$ lie in the same SCC, $\forall u \in V$

PROOF: $u \mapsto \phi(u)$ (by definition) and $\phi(u) \mapsto u$ since $\phi(u)$ is an ancestor of u (by THM2)

THM3: two vertices $u,v \in V$ lie in the same SCC $\Leftrightarrow \phi(u) = \phi(v)$ in a DFS of $G = (V, E)$

PROOF: let u and v be in the same SCC $C_{uv} \Rightarrow u \leftrightarrows v$

Strongly Connected Components



$\forall w$: $v \mapsto w \Rightarrow u \mapsto w$ and $\forall w$: $u \mapsto w \Rightarrow v \mapsto w$, i.e.,
every vertex reachable from u is reachable from v and vice-versa

So, $w = \phi(u) \Rightarrow w = \phi(v)$ and $w = \phi(v) \Rightarrow w = \phi(u)$ by definition of forefather

PROOF: Let $\phi(u) = \phi(v) = w \in C_w \Rightarrow u \in C_w$ by C1 and $v \in C_w$ by C1

By THM3: SCCs are sets of vertices with the same forefather

By THM2 and parenthesis THM: A forefather is the first vertex discovered and the last vertex finished in its SCC

SCC: Why do we Run DFS on GT?

Consider $r \in V$ with largest finishing time computed by DFS on G

r must be a forefather by definition since $r \mapsto r$ and $f[r]$ is maximum in V

C_r = ?: C_r = vertices in r's SCC = $\{u \text{ in } V: \phi(u) = r\}$

$\Rightarrow C_r = \{u \in V: u \mapsto r \text{ and } f[x] \le f[r]\ \forall x \in R_u\}$ where $R_u = \{v \in V: u \mapsto v\}$

$\Rightarrow C_r = \{u \in V: u \mapsto r\}$ since $f[r]$ is maximum

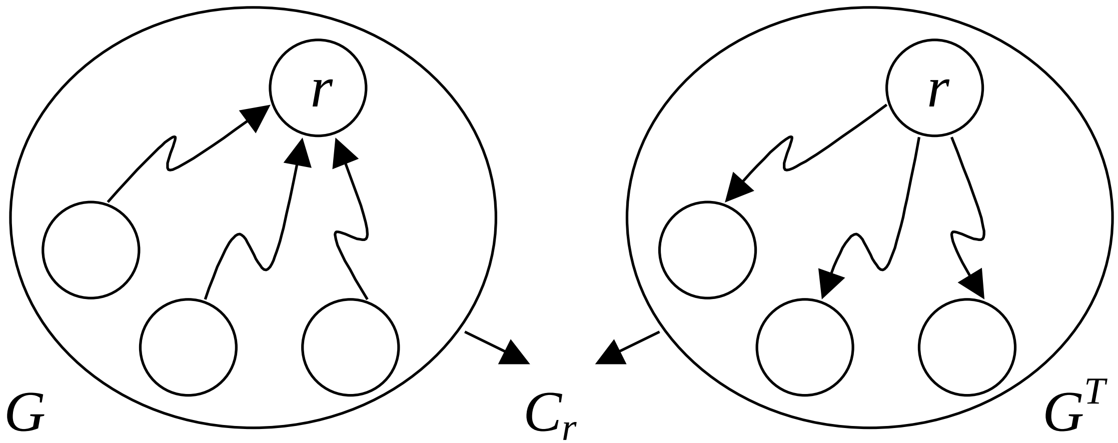
$\Rightarrow C_r = R_r^T = \{u \in V: r \mapsto u \text{ in } G^T\}$ = reachability set of r in G^T

i.e., C_r = those vertices reachable from r in G^T

Thus DFS-VISIT(G^T, r) identifies all vertices in C_r and blackens them

SCC: Why do we Run DFS on GT?

BFS(G^T, r) can also be used to identify C_r

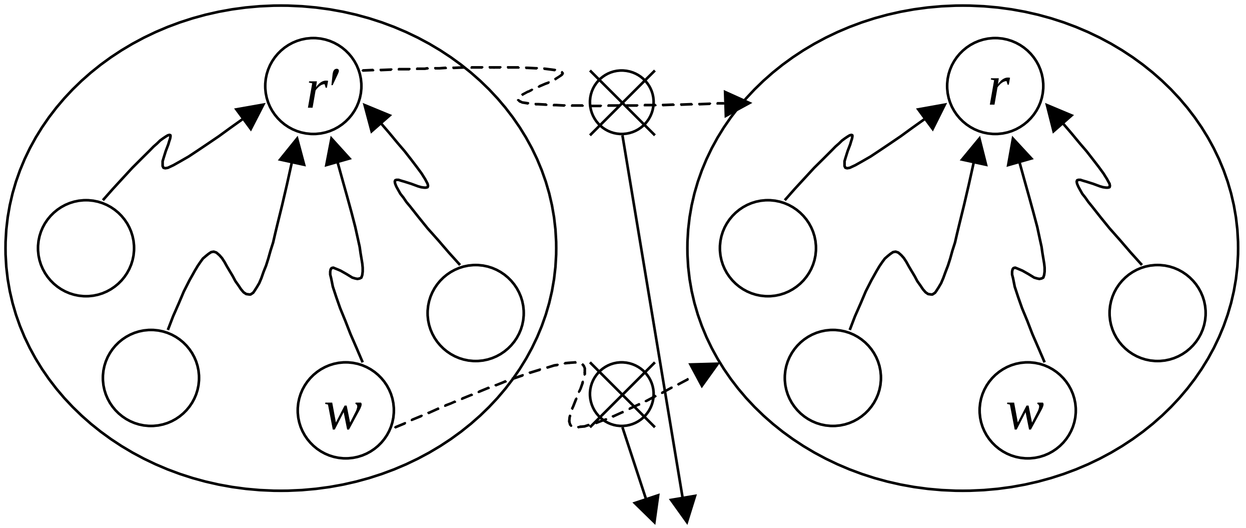


Then, DFS on G^T continues with DFS-VISIT(G^T, r')
where $f[r'] > f[w]\ \forall w \in V - C_r$

r must be a forefather by definition since $r' \mapsto r'$ and
$f[r']$ is maximum in $V - C_r$

SCC: Why do we Run DFS on GT?

Hence by similar reasoning DFS-VISIT(G^T, r') identifies $C_{r'}$



Impossible since otherwise

$r', w \in C_r \Rightarrow r', w$ would have been blackened

Thus, each DFS-VISIT(G^T, x) in DFS(G^T)

identifies an SCC C_x with $\phi = x$