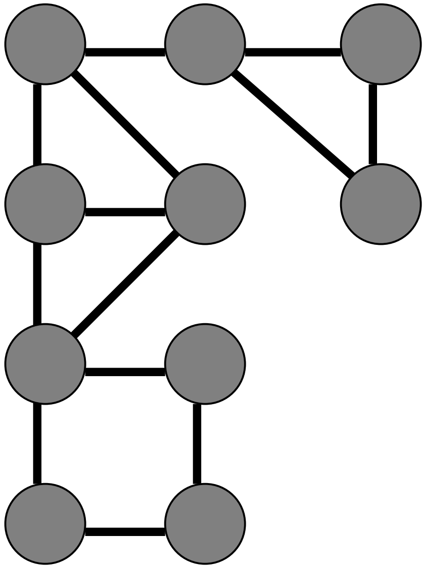
Articulation Point

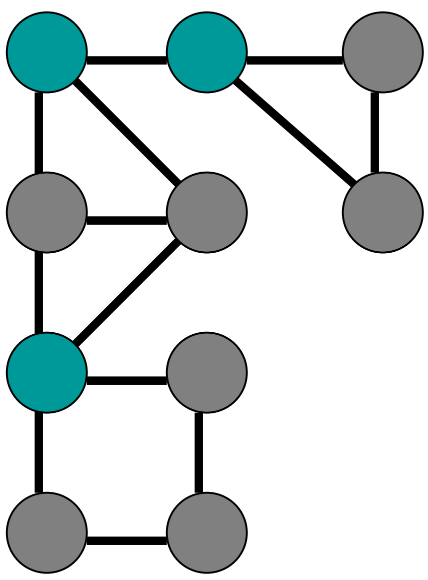
- Let G = (V,E) be a connected undirected graph.

Articulation Point: is any vertex of G whose removal results in a disconnected graph.



Articulation Point

Articulation Point: is any vertex of G whose removal results in a disconnected graph.



Biconnected Components

- Let G = (V, E) be a connected, undirected graph.
- An articulation point of G is a vertex whose removal disconnects G.
- A biconnected component of G is a maximal set of edges such that any two edges in the set lie on a common simple cycle.
- A **biconnected component** of a graph is a connected subgraph that cannot be broken into disconnected pieces by deleting any single node.
- We can say that a graph G is a bi-connected graph if it is
    1. Connected (it is possible to reach every vertex from every other vertex, by a simple path)
    2. There are no articulation points.

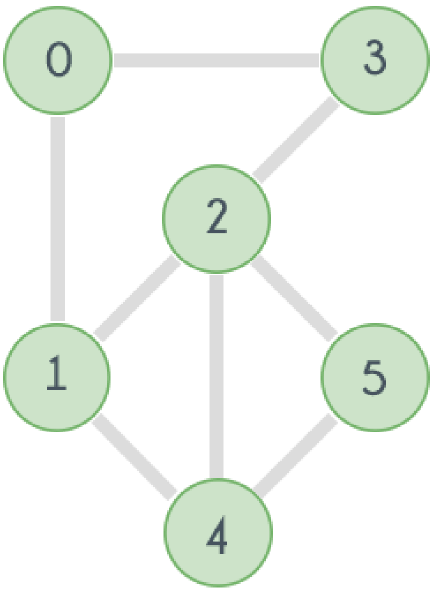


Fig. 1

- Now try removing the vertices one by one and observe. Removing any of the vertices does not increase the number of connected components. So the given graph is Biconnected.

- Now consider the following graph which is a slight modification in the previous graph.

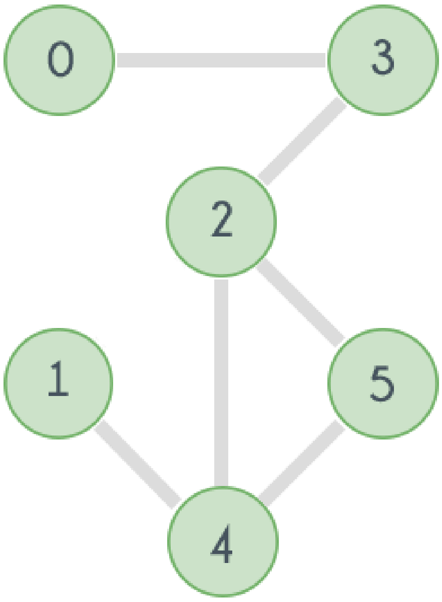


Fig. 2

- In the above graph if the vertex 2 is removed, then here's how it will look:

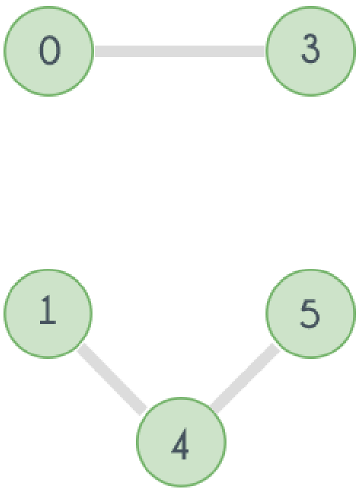
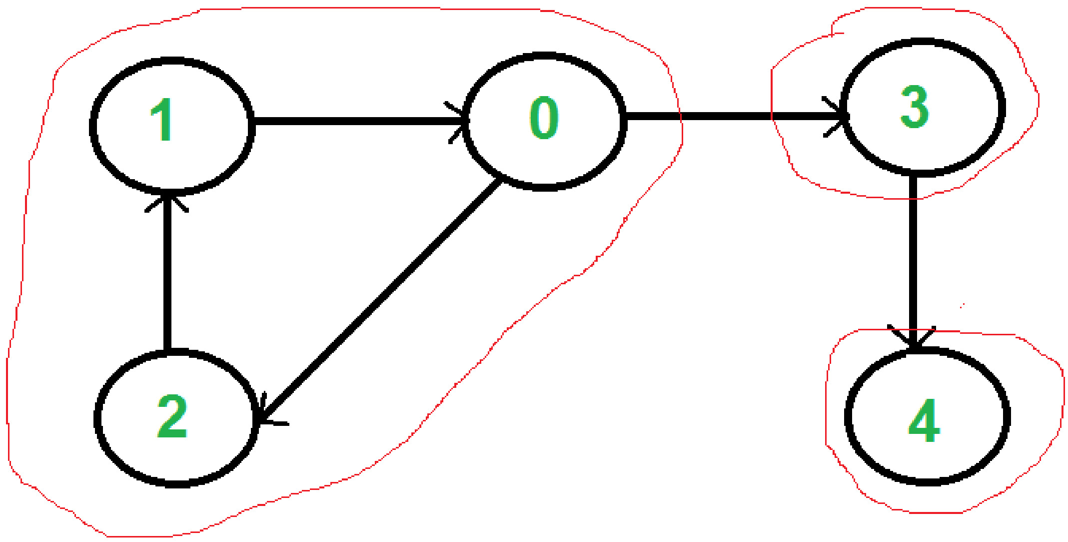


Fig. 3

Strongly Connected Components

- A directed graph is strongly connected if there is a path between all pairs of vertices.
- A strongly connected component (**SCC**) of a directed graph is a maximal strongly connected subgraph.
- For example, there are 3 SCCs in the following graph



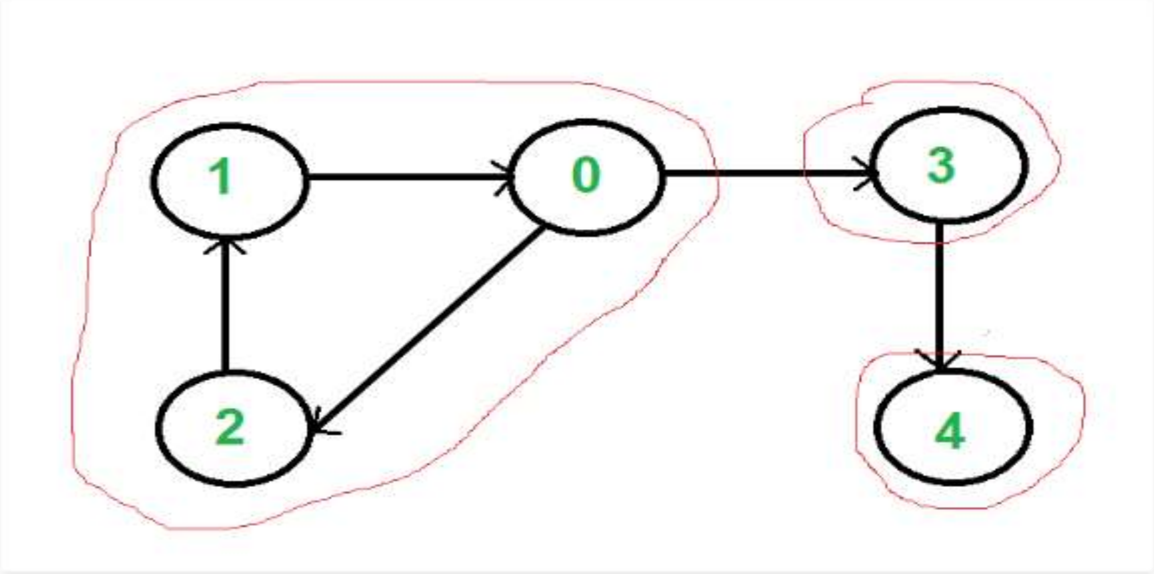
SCC1-{0,1,2}

SCC2-{3}

SCC3-{4}

Strongly Connected Components

- A directed graph is strongly connected if there is a path between all pairs of vertices.
- A strongly connected component (**SCC**) of a directed graph is a maximal strongly connected subgraph.



Strongly Connected Components

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

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

- 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

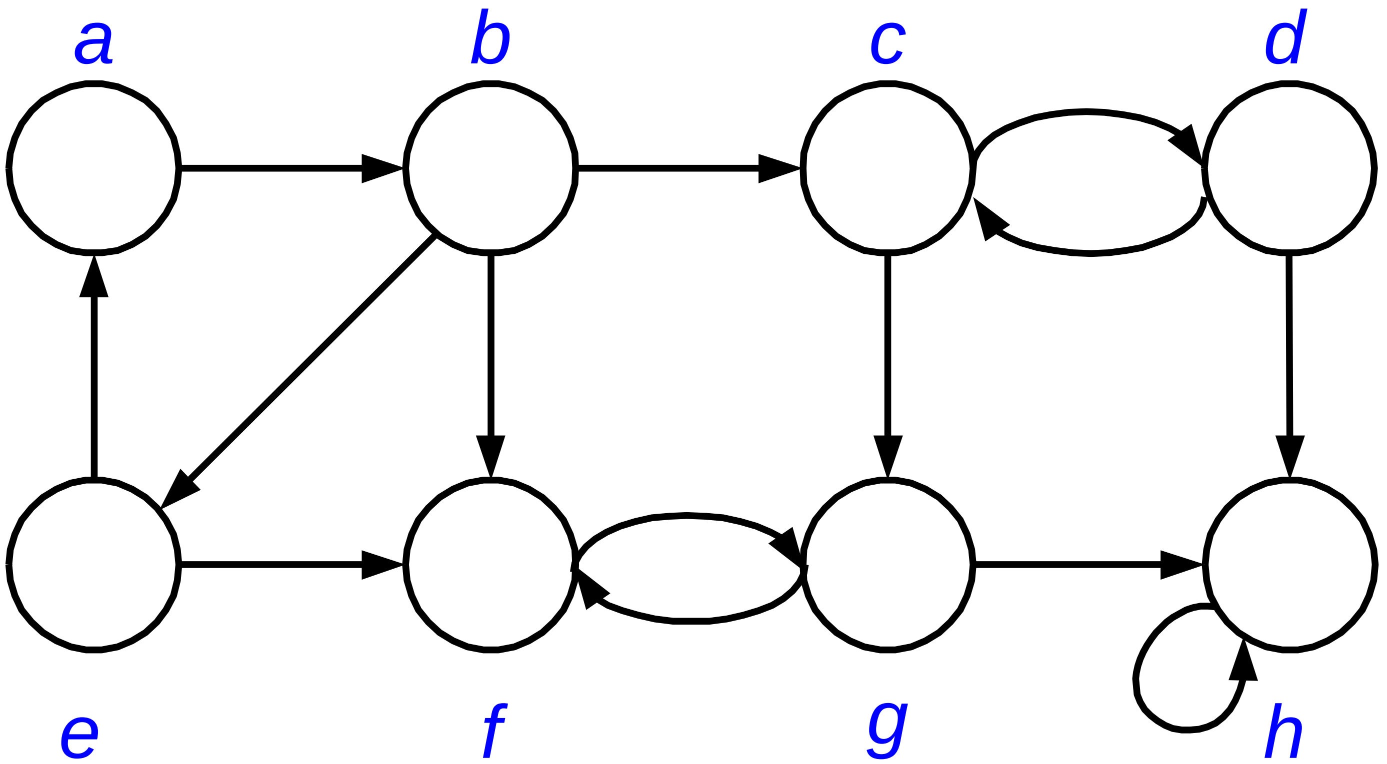
(1) Run **DFS**(G) to compute finishing times for all $u \in V$

(2) Compute G^T

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

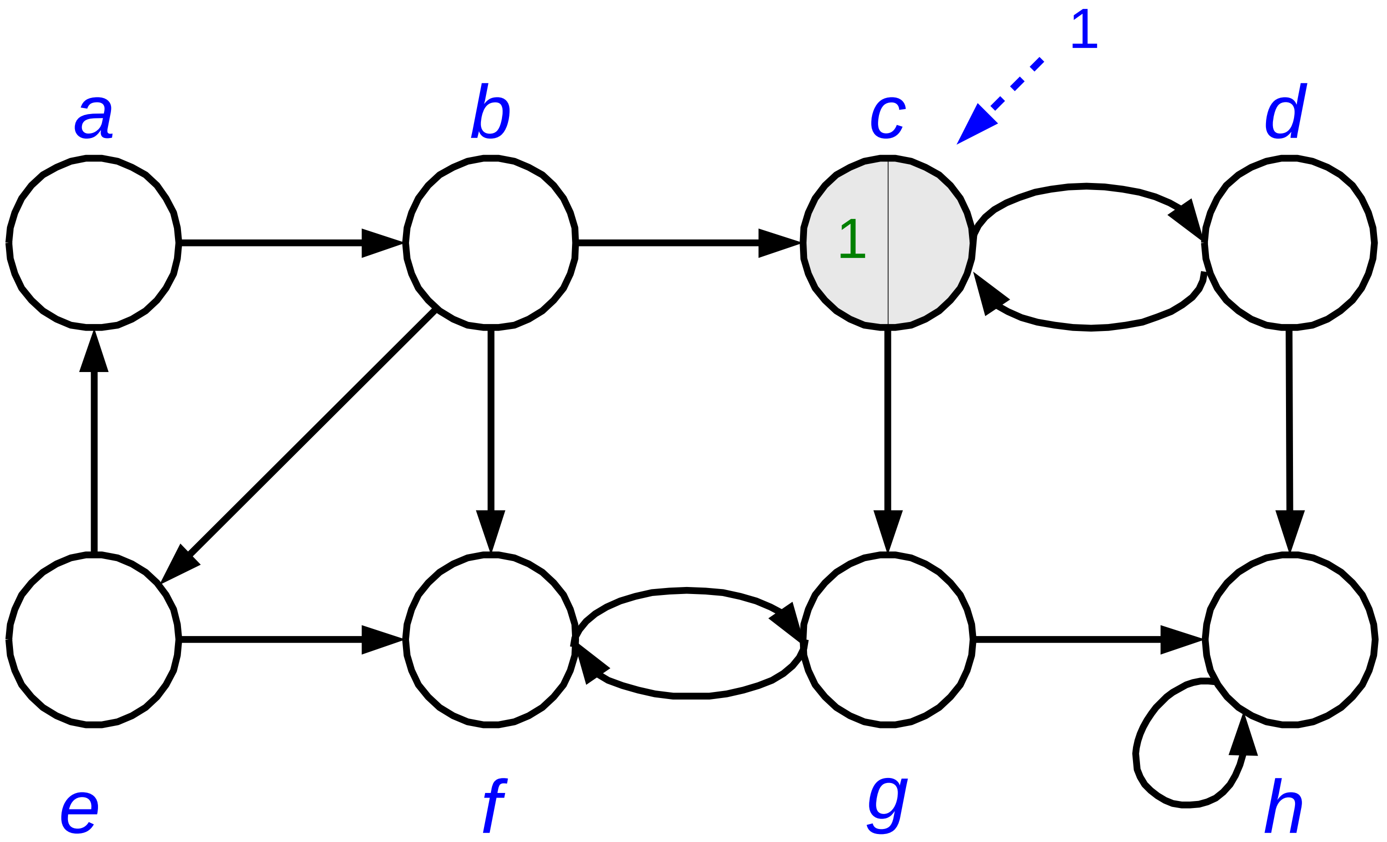
(4) Output vertices of each DFT in DFF of Step (3) as a separate SCC

SCC: Example



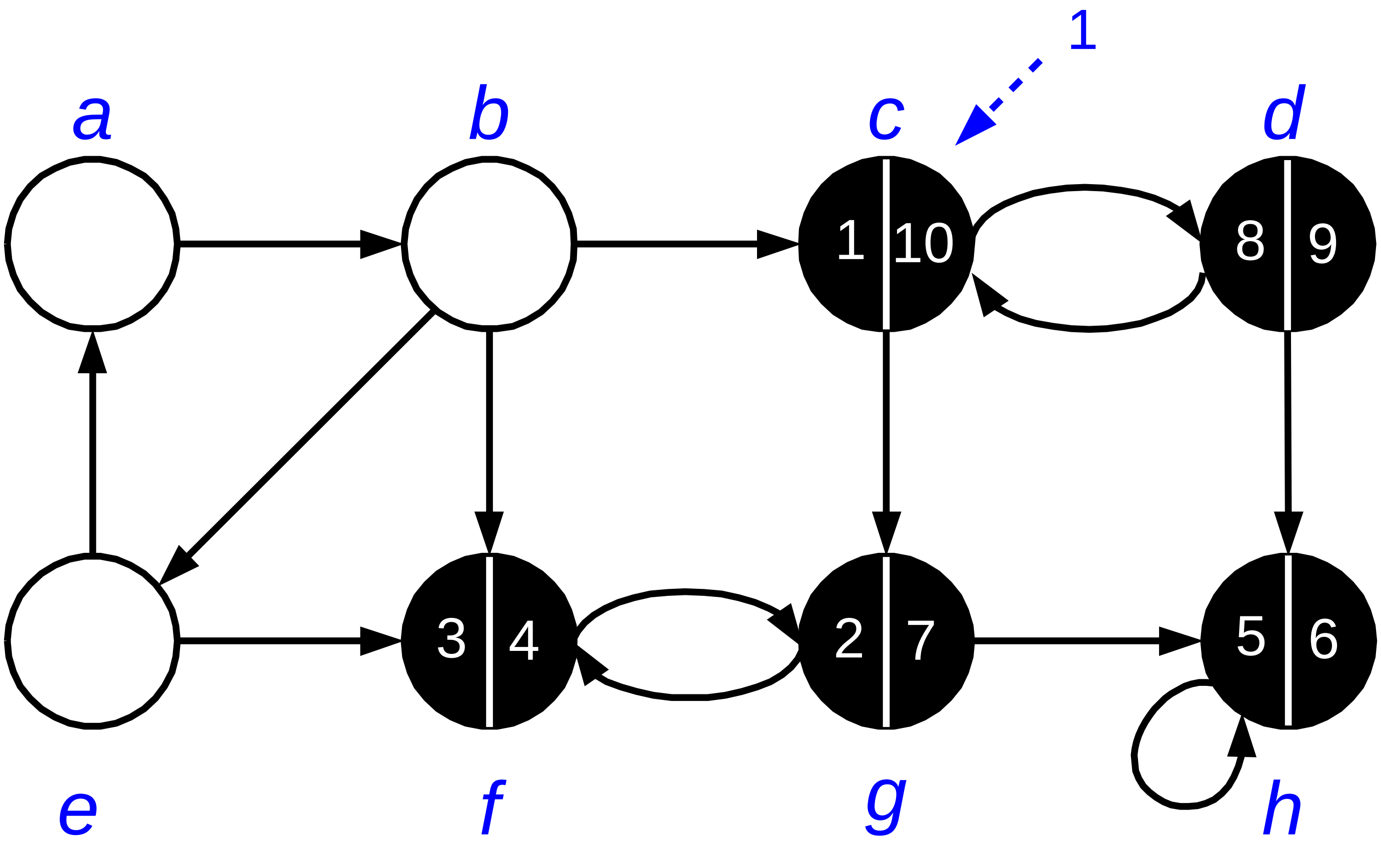
SCC: Example

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



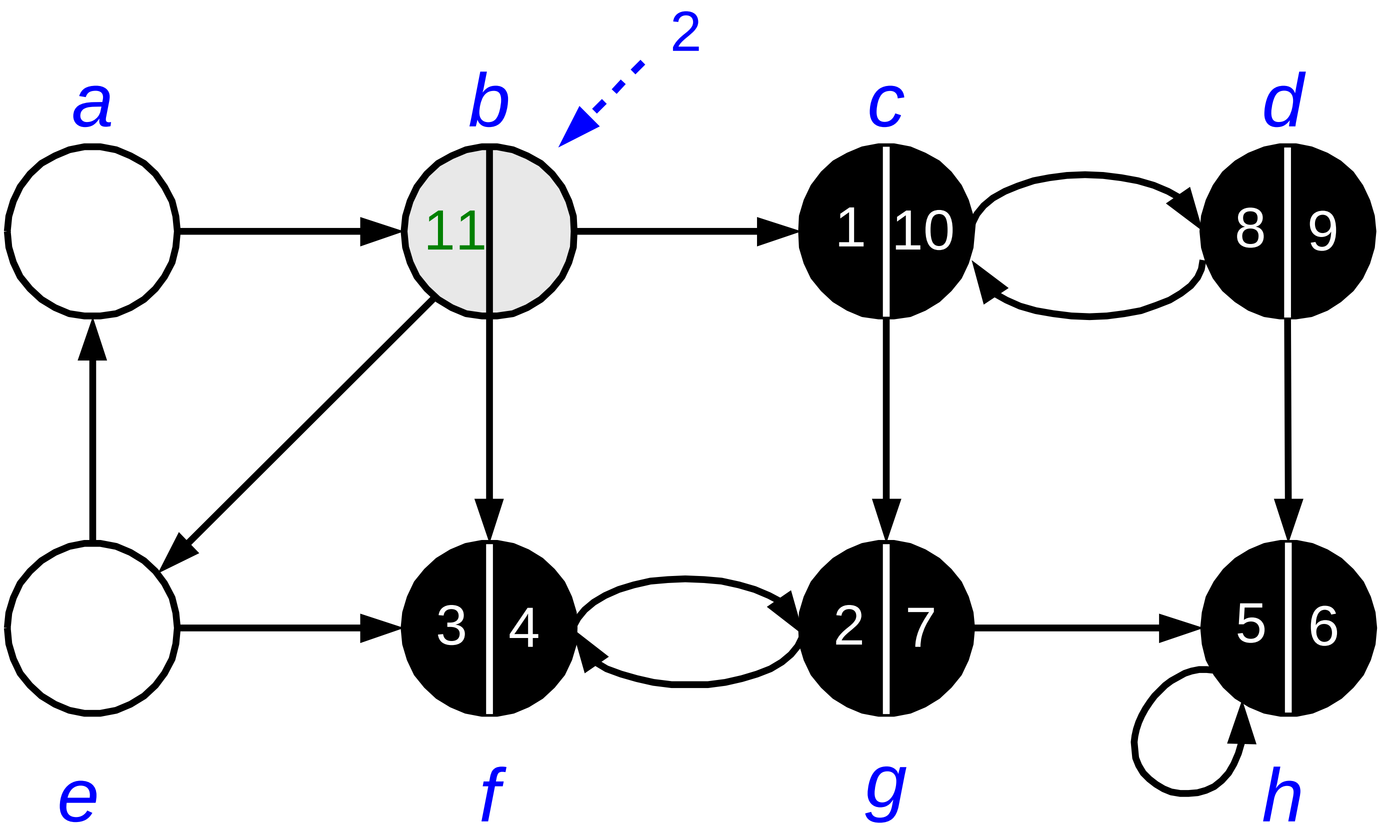
SCC: Example

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

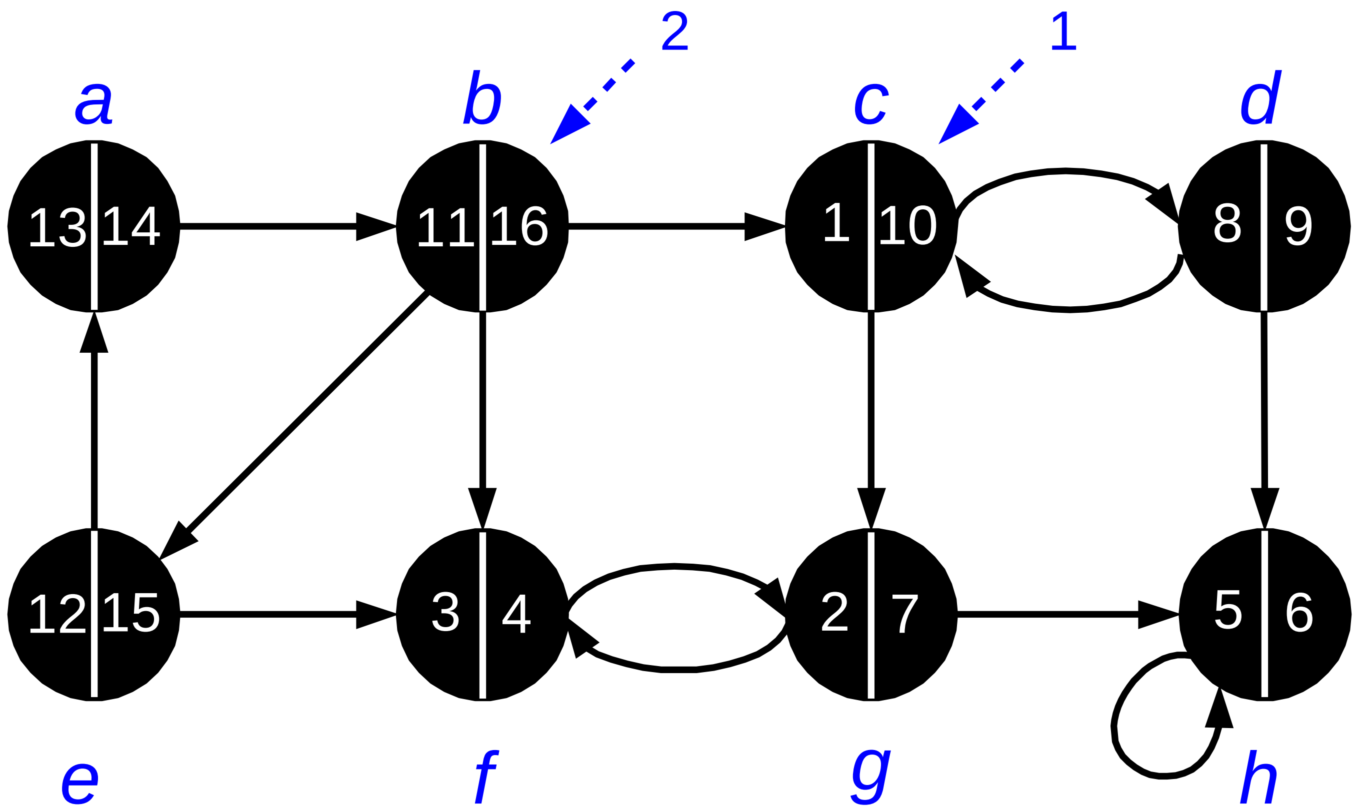


SCC: Example

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



SCC: Example

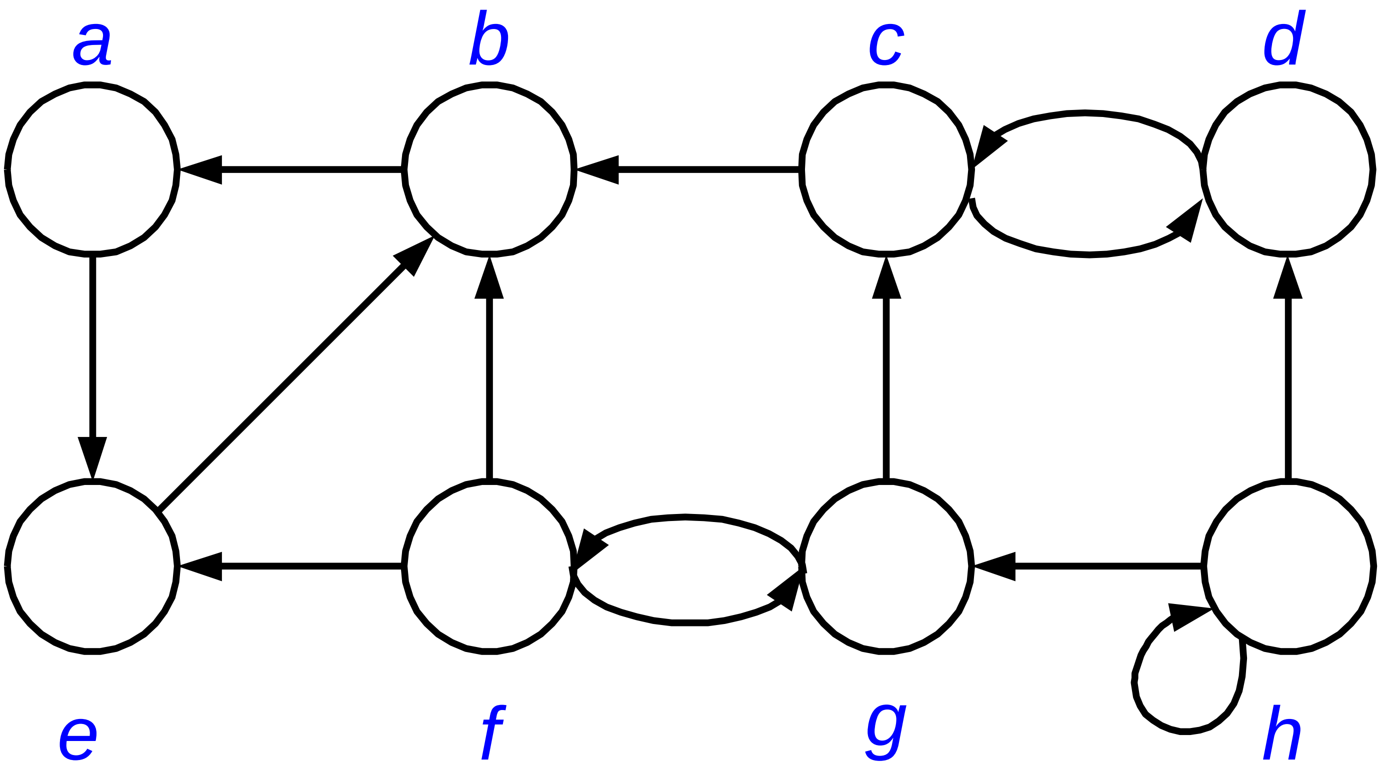


Vertices sorted according to the finishing times:

$\langle$***b***, ***e***, ***a***, ***c***, ***d***, ***g***, ***h***, ***f*** $\rangle$

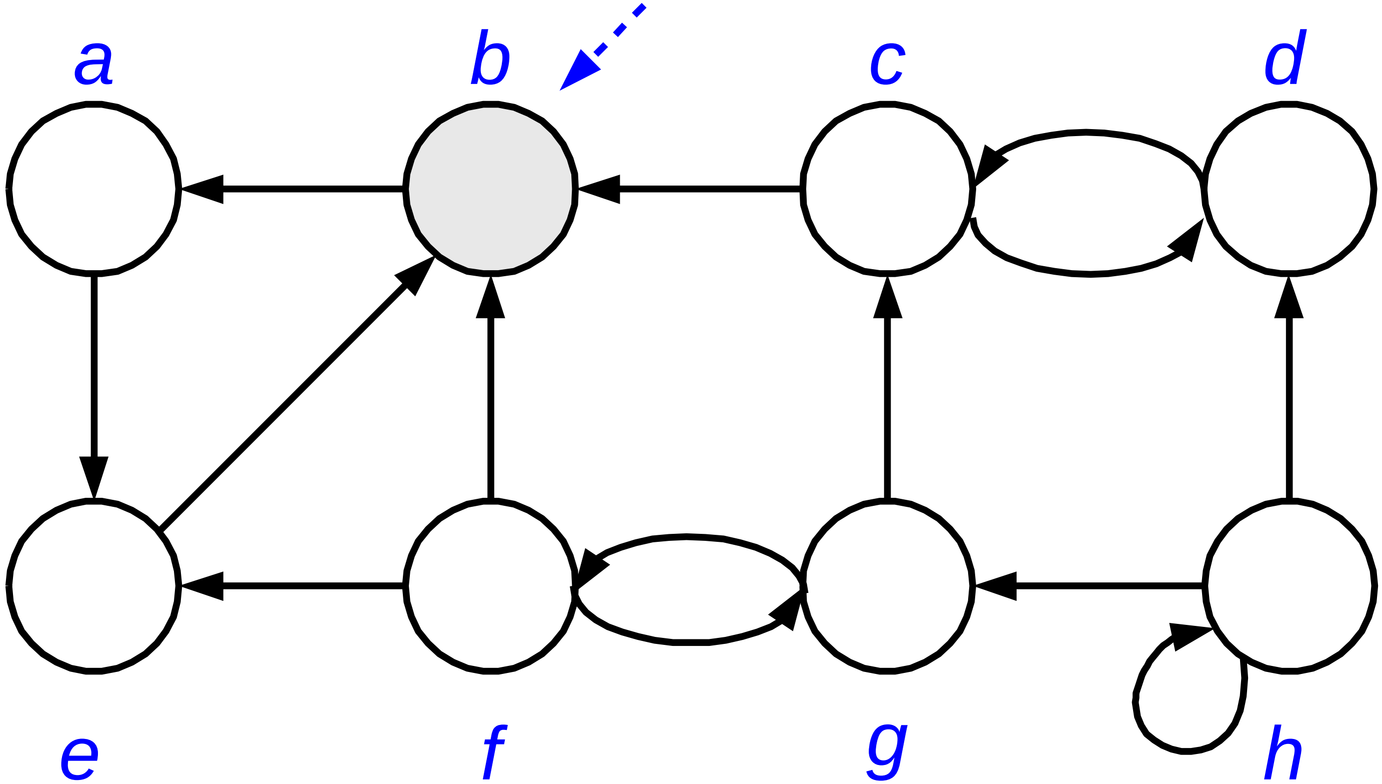
SCC: Example

(2) Compute G^T



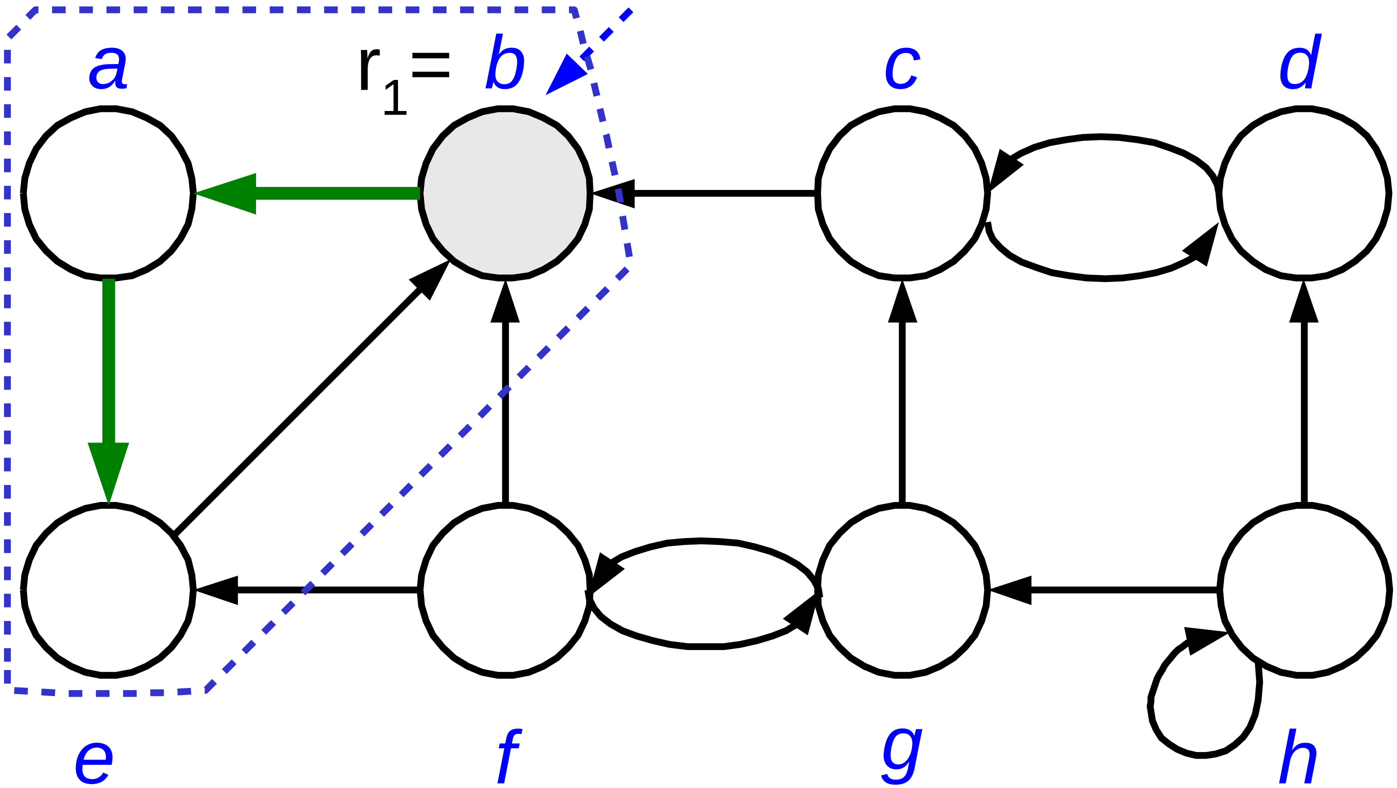
SCC: Example

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



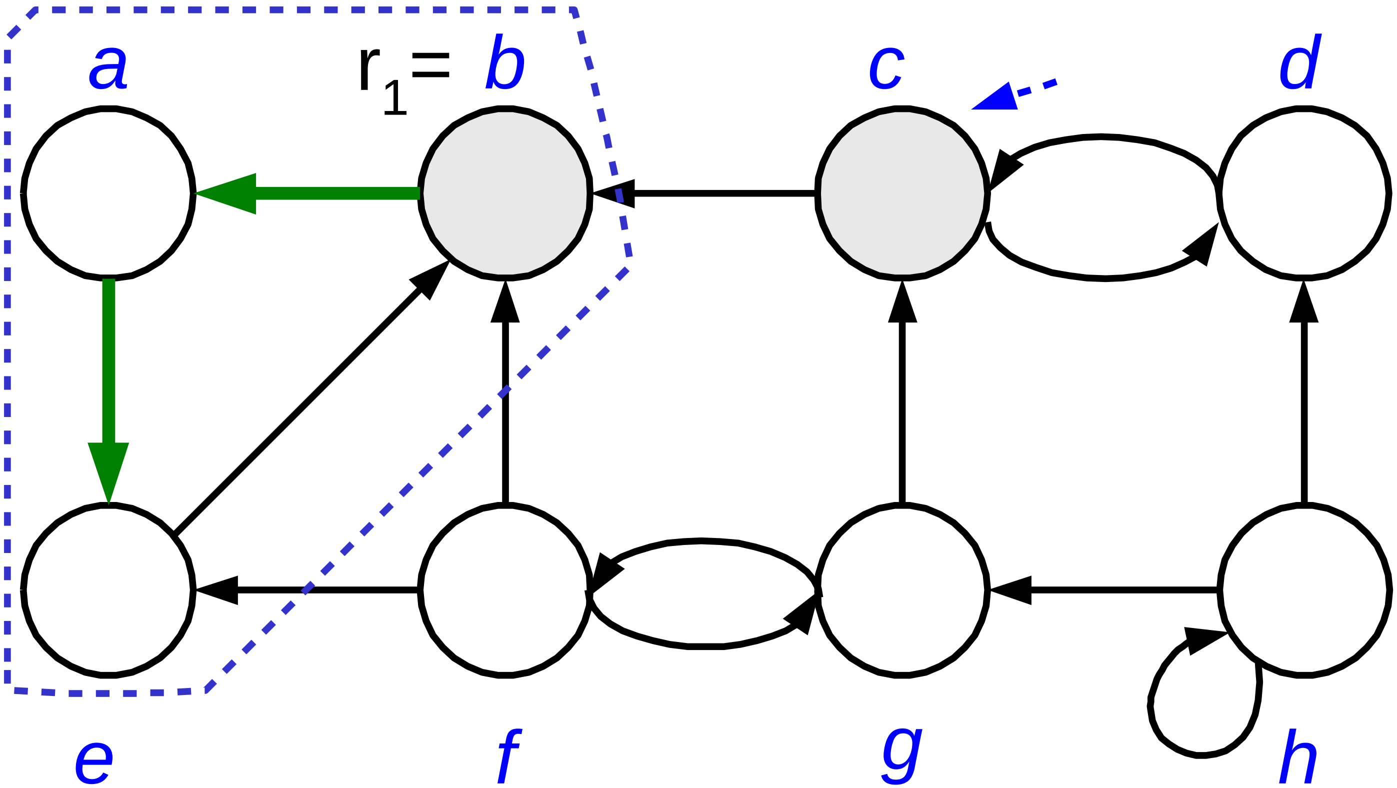
SCC: Example

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



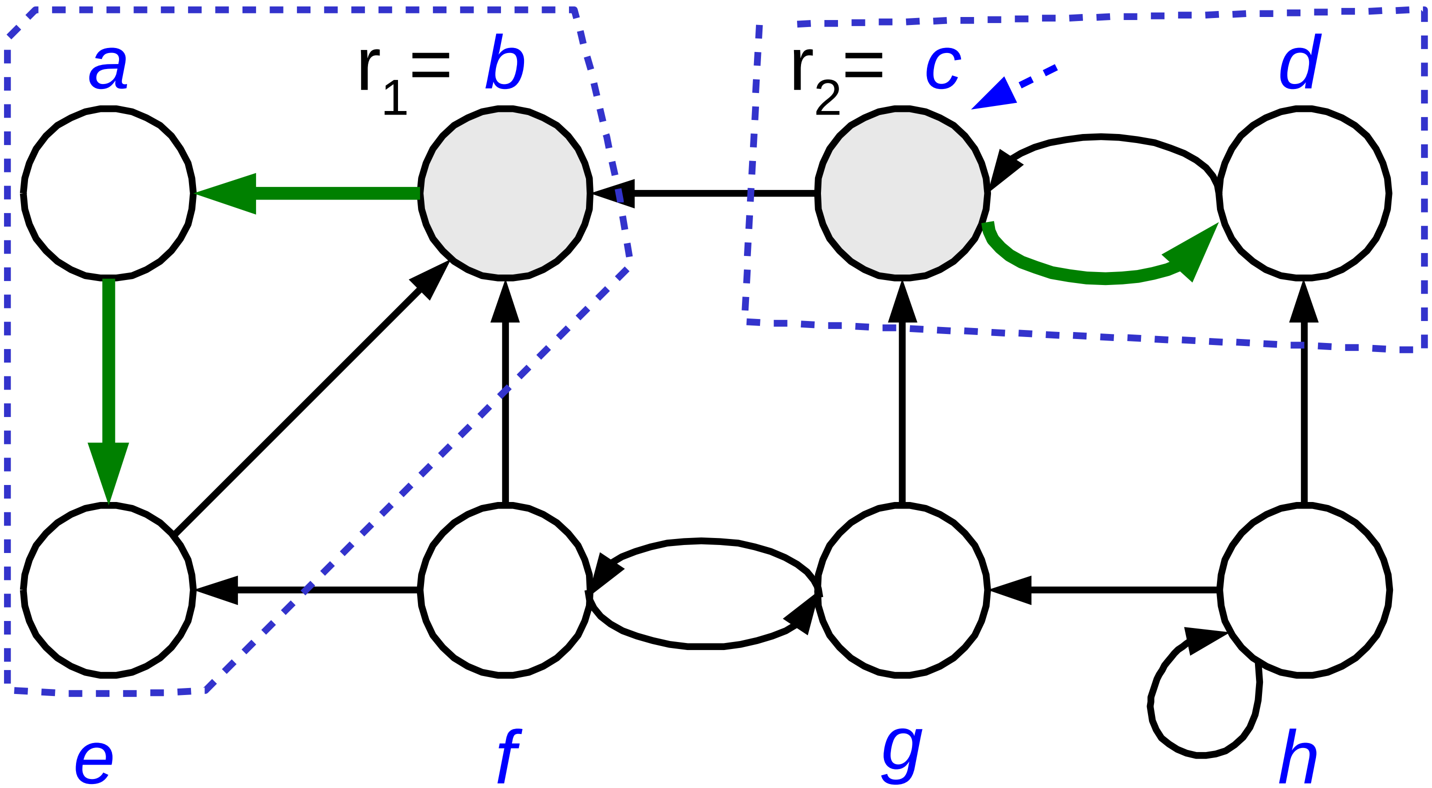
SCC: Example

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



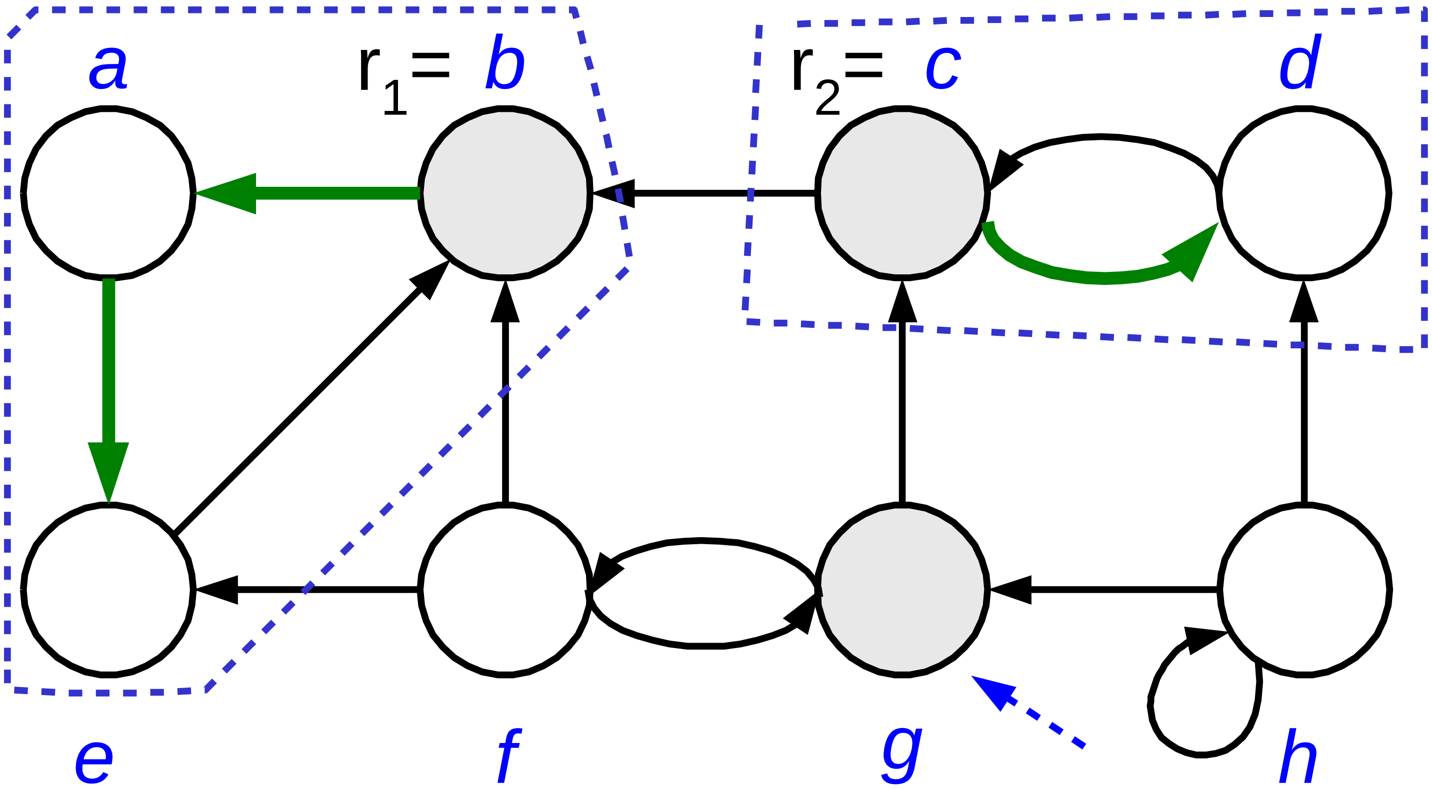
SCC: Example

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



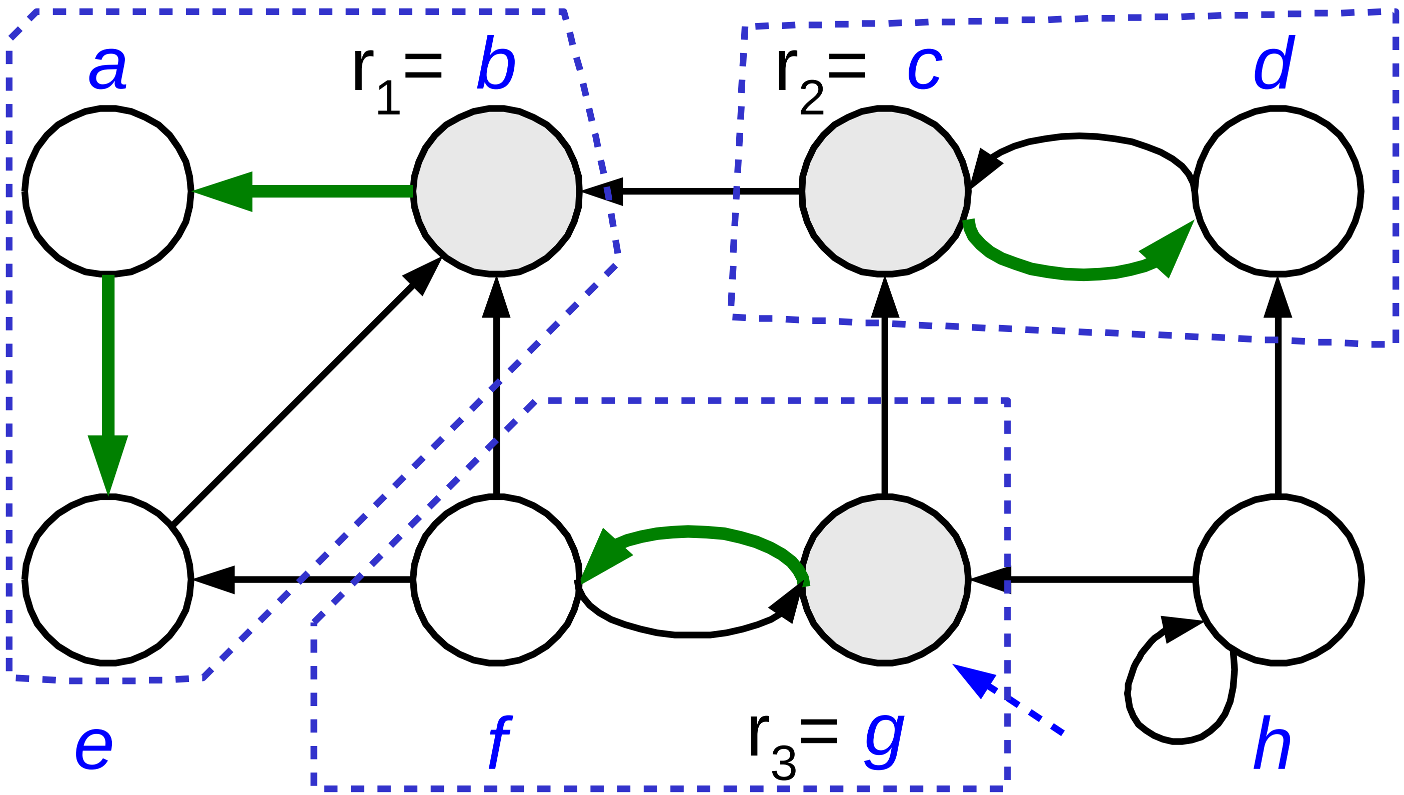
SCC: Example

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



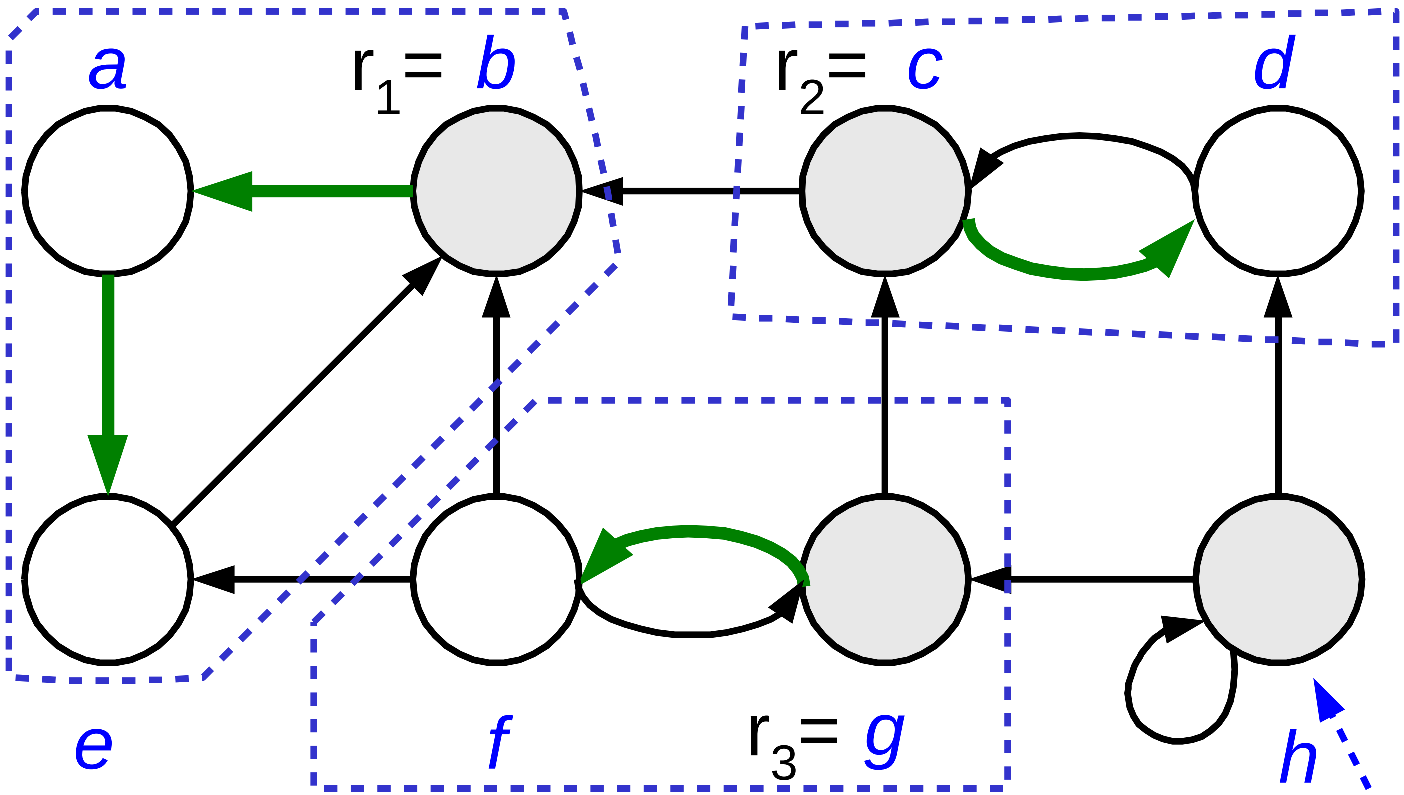
SCC: Example

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



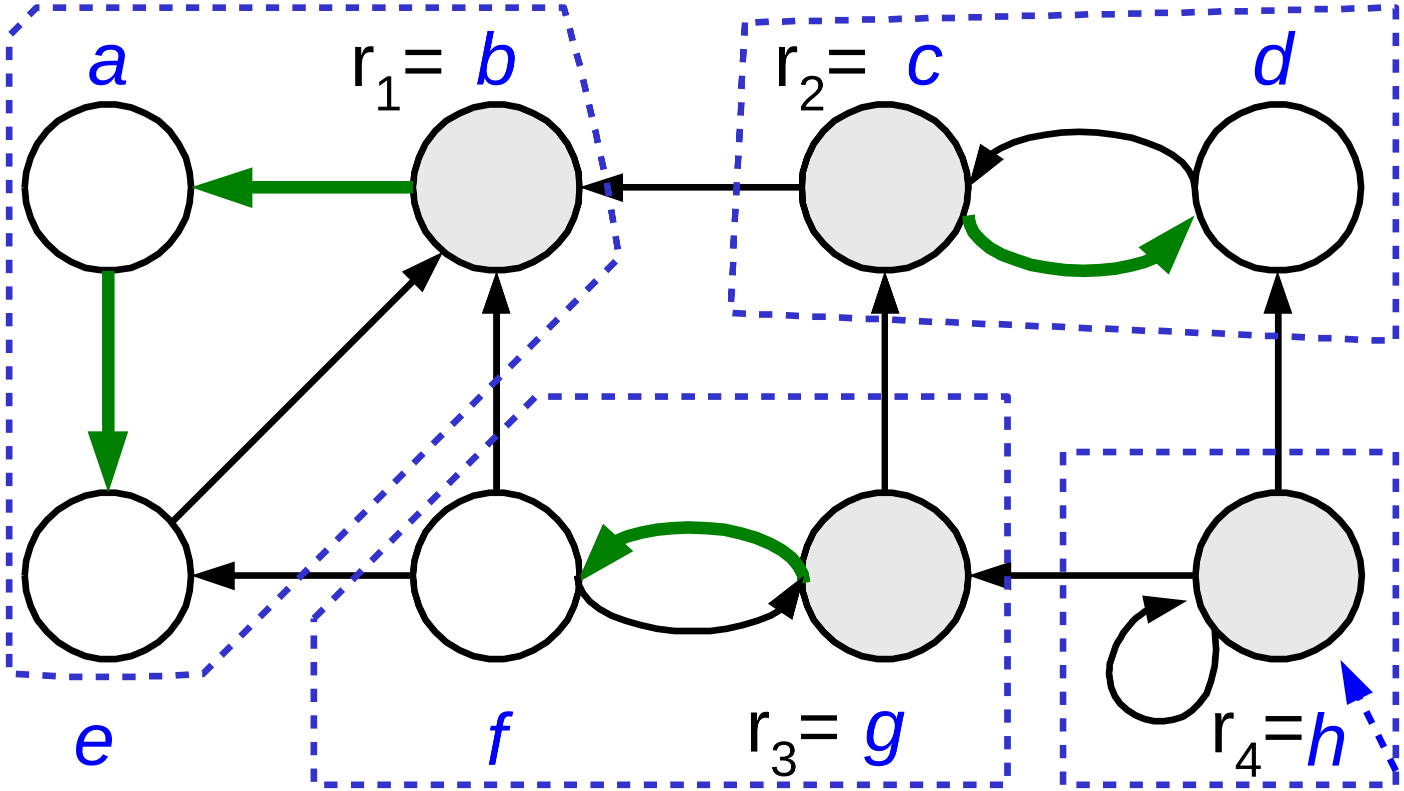
SCC: Example

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



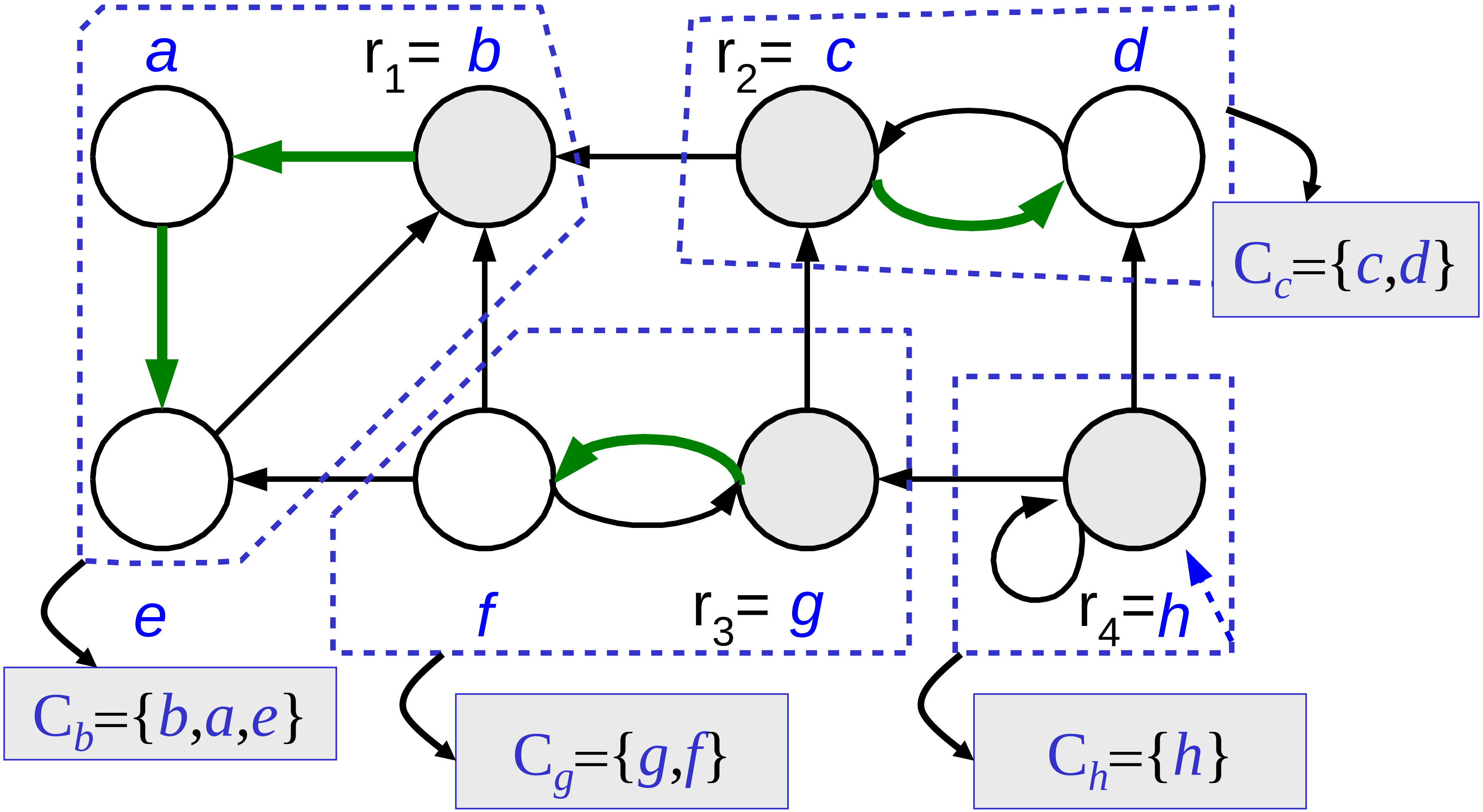
SCC: Example

(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