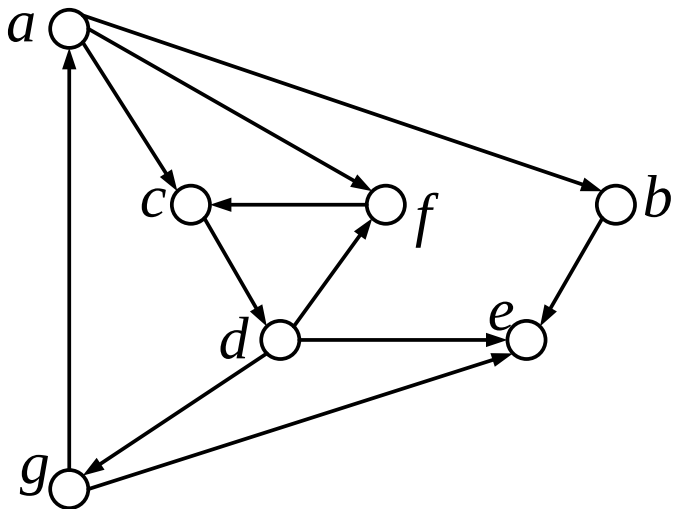
Additional Discussions on DFS

Yufei Tao's Teaching Team

This tutorial will provide extra examples to enhance your understanding of several crucial properties of the DFS algorithm.

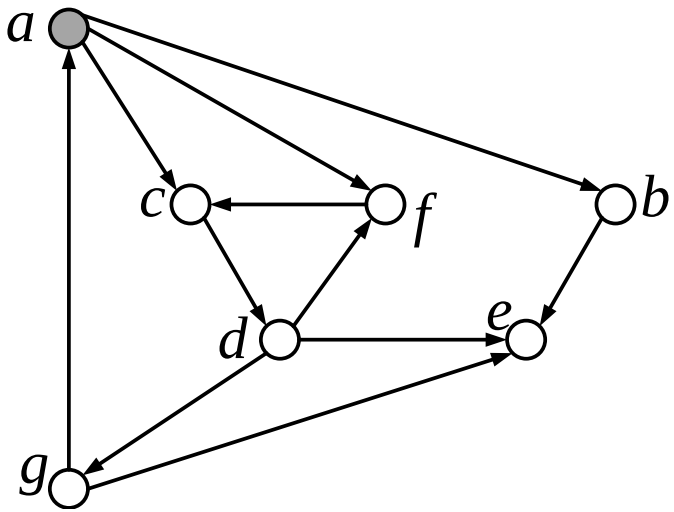
Input



Suppose we start from the vertex a, namely a is the root of DFS tree.

DFS

First, color all the vertices white. Then, create a **stack** S, push the starting vertex a into S and color it gray. Create a DFS tree with a as the root and set its time interval to $I(a) = [1, -]$.

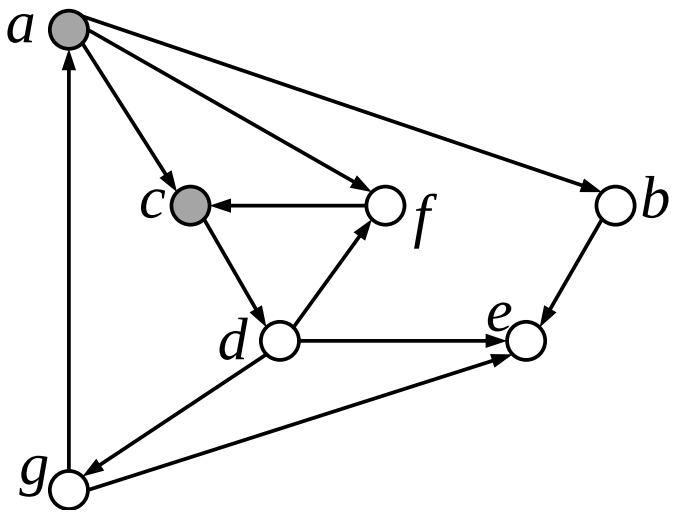


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$

$S = (a)$.

DFS

Top of stack: a, which has white out-neighbors b, c, f. Suppose we access c first. Push c into S.

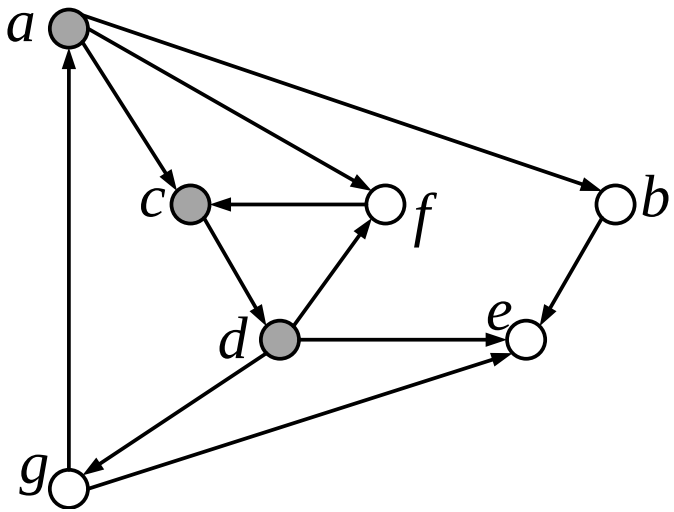


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$
|	
c	$I(c) = [2, \ ]$

$S = (a, c)$.

DFS

After pushing d into S:

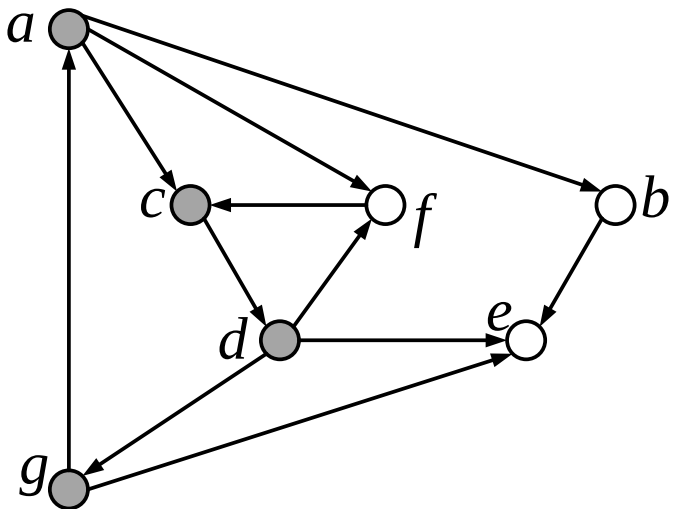


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$
|	
c	$I(c) = [2, \ ]$
|	
d	$I(d) = [3, \ ]$

$S = (a, c, d)$.

DFS

Now d tops the stack. It has white out-neighbors e, f and g. Suppose we visit g first. Push g into S.



DFS Tree

a
|
c
|
d
|
g

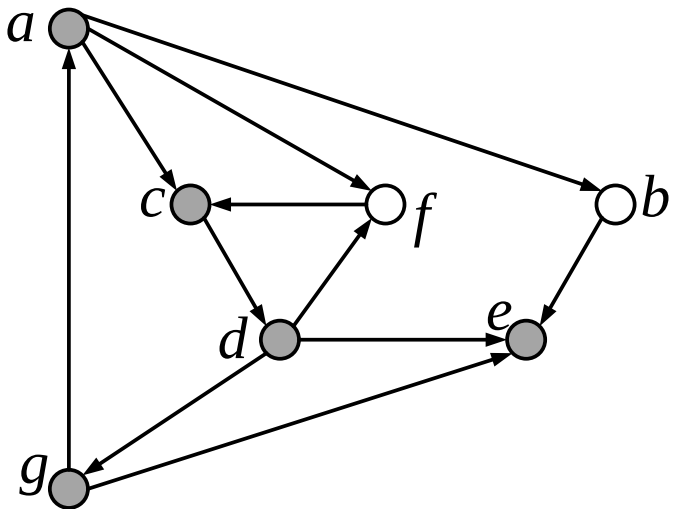
Time Interval

$I(a) = [1, \ ]$
$I(c) = [2, \ ]$
$I(d) = [3, \ ]$
$I(g) = [4, \ ]$

$S = (a, c, d, g)$.

DFS

After pushing e into S:

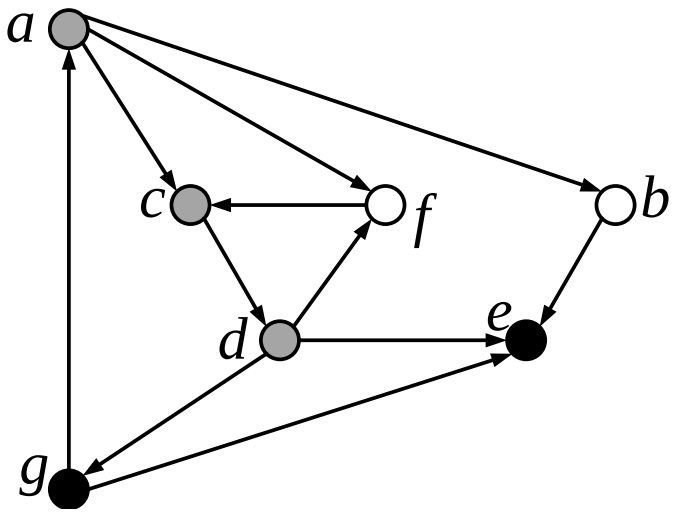


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$
$\vert$	
c	$I(c) = [2, \ ]$
$\vert$	
d	$I(d) = [3, \ ]$
$\vert$	
g	$I(g) = [4, \ ]$
$\vert$	
e	$I(e) = [5, \ ]$

$S = (a, c, d, g, e)$.

DFS

e has no white out-neighbors. So pop it from S and color it black.
Similarly, g has no white out-neighbors. Pop it from S and color it black.

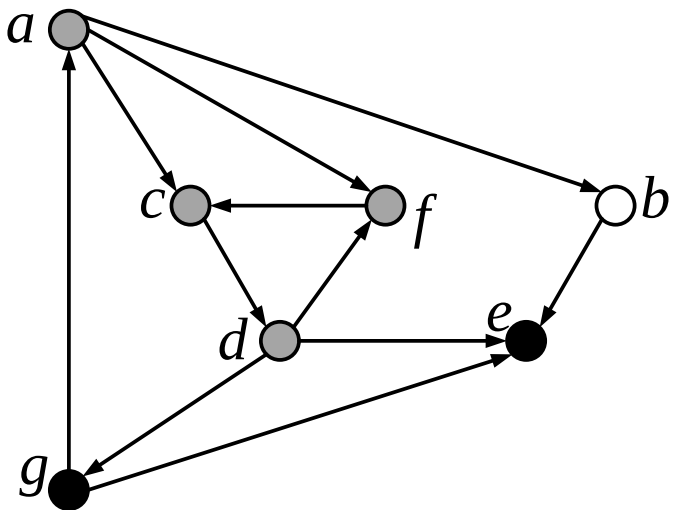


DFS Tree	Time Interval
a	$I(a) = [1, \ ]$
$\vert$	
c	$I(c) = [2, \ ]$
$\vert$	
d	$I(d) = [3, \ ]$
$\vert$	
g	$I(g) = [4, 7]$
$\vert$	
e	$I(e) = [5, 6]$

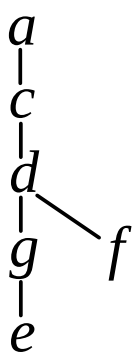
$S = (a, c, d)$.

DFS

Now d tops the stack again. It still has a white out-neighbor f. So, push f into S.



DFS Tree



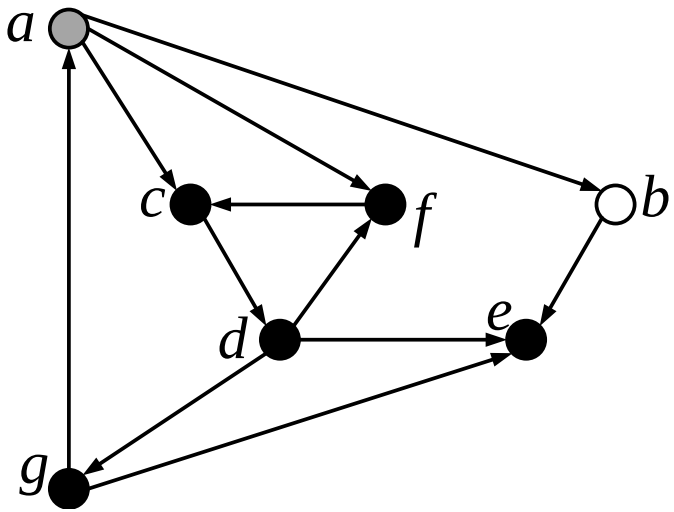
Time Interval

$I(a) = [1, \ ]$
$I(c) = [2, \ ]$
$I(d) = [3, \ ]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, \ ]$

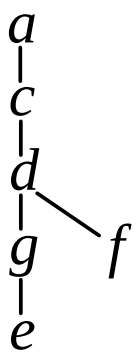
$S = (a, c, d, f)$.

DFS

After popping f, d, c:



DFS Tree



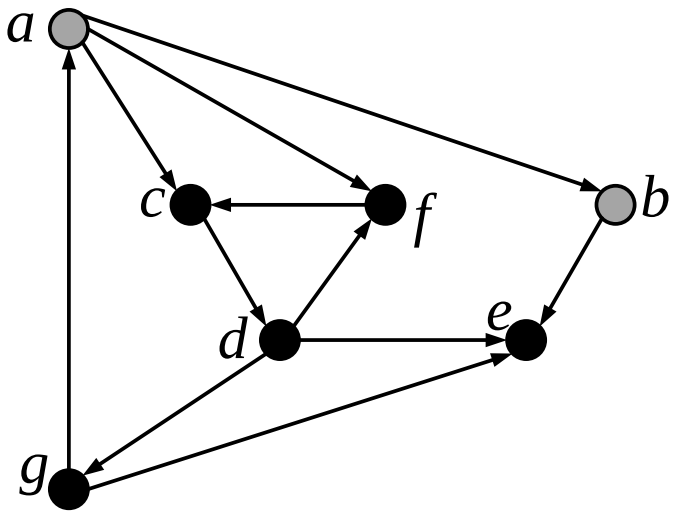
Time Interval

$I(a) = [1, \ ]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$

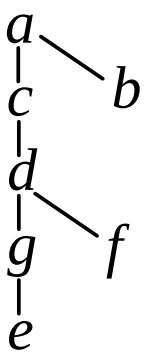
$S = (a)$.

DFS

Now a tops the stack again. It still has a white out-neighbor b. So, push b into S.



DFS Tree



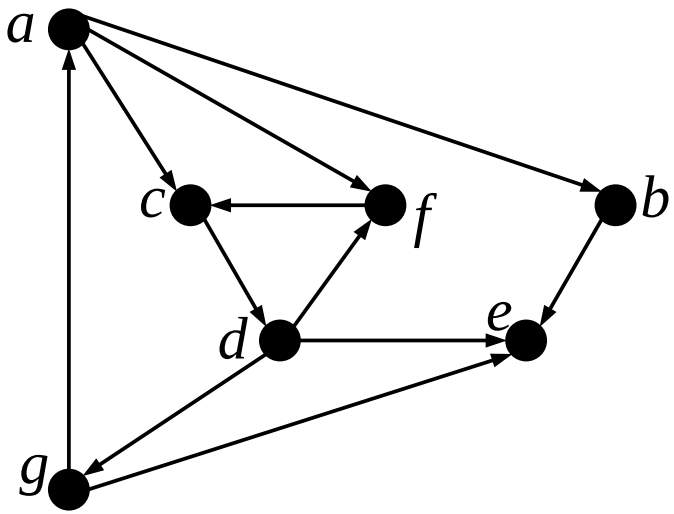
Time Interval

$I(a) = [1, \ ]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, \ ]$

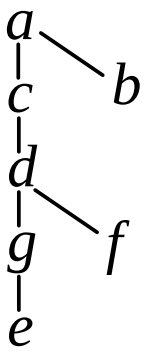
$S = (a, b)$.

DFS

After popping b and a:



DFS Tree



Time Interval

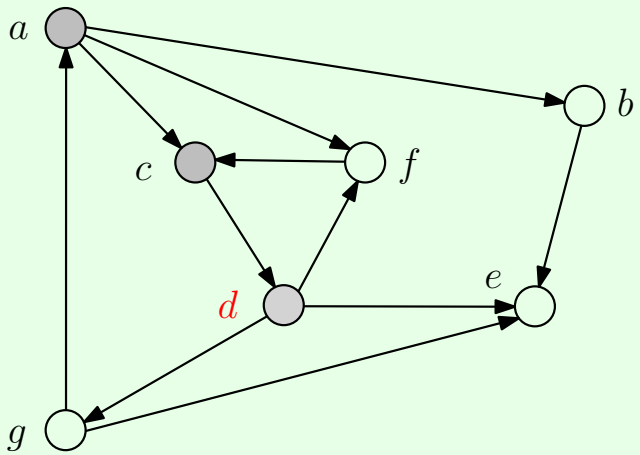
$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

$S = ()$.

There are no more white vertices. The algorithm terminates.

Lemma (the Ancestor-Descendent Lemma): Let u and v be two distinct vertices in G. Then, u is an ancestor of v in the DFS-forest **if and only if** the following holds: u is already in the stack when v enters the stack.

Example: When d enters the stack, a and c are in the stack. d is a descendant of a and c in the DFS-tree.



DFS Tree

a
c
d

Time Interval

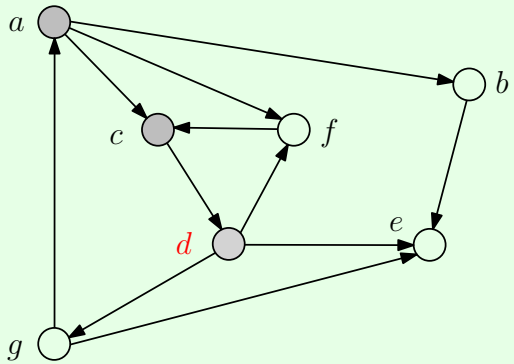
$I(a) = [1, \ ]$
$I(c) = [2, \ ]$
$I(d) = [3, \ ]$

$S =$ | a c d |

Theorem (the Parenthesis Theorem): Let u and v be two distinct vertices in G. Then:

- $I(u)$ contains $I(v)$ **if and only if** u is an ancestor of v in the DFS-forest.
- $I(u)$ and $I(v)$ are disjoint **if and only if** neither u nor v is an ancestor of the other.

Example: When d enters the stack, a and c are in the stack. d is a descendant of a and c in the DFS-tree.



DFS Tree

Time Interval

$I(a) = [1, \ ]$

$I(c) = [2, \ ]$

$I(d) = [3, \ ]$

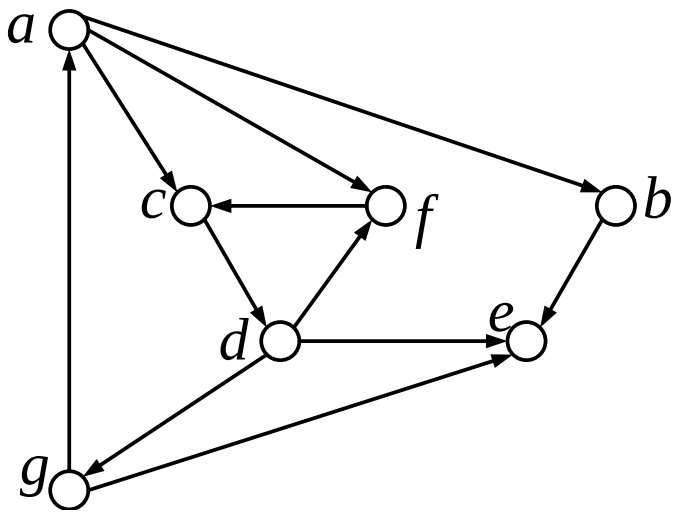
$S =$ $\boxed{a \ c \ d}$

DFS can be used to solve many classicial problems in computer science. In this course, we will see two of those problems. The first one is cycle detection, as we discuss next.

Cycle Detection

Problem Input:

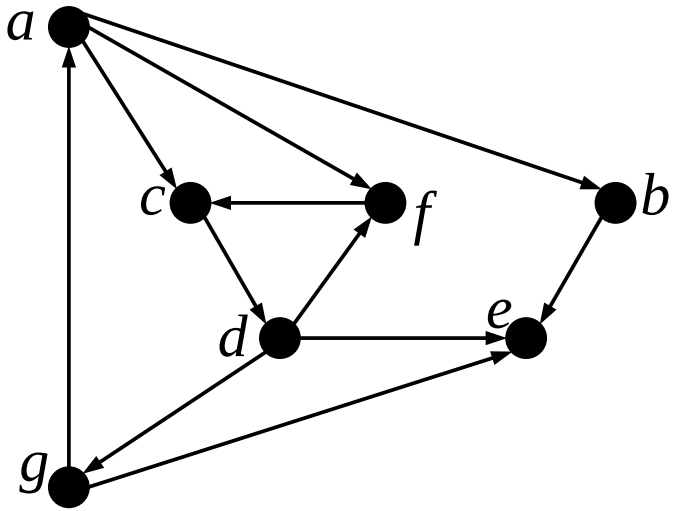
A directed graph.



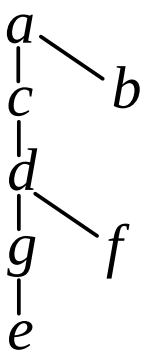
Problem Output:

A boolean value indicating whether the graph contains a cycle.

First Step: DFS



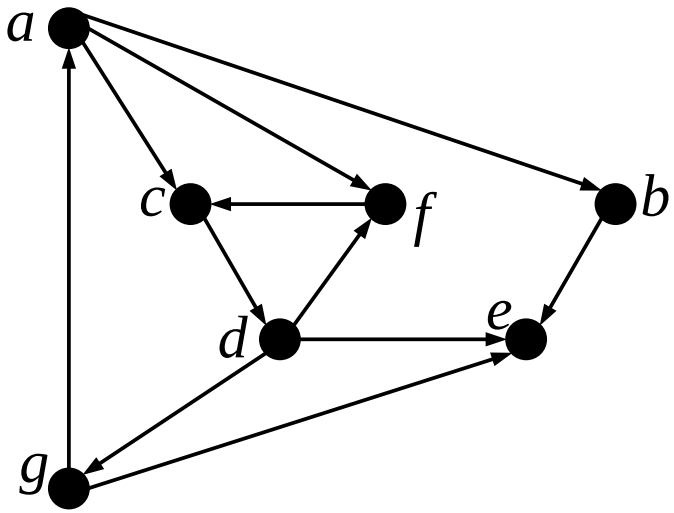
DFS Tree



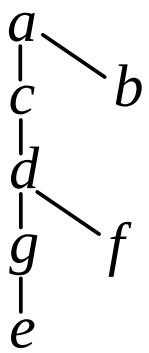
Time Interval

$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

Second Step: Find Back Edges



DFS Tree



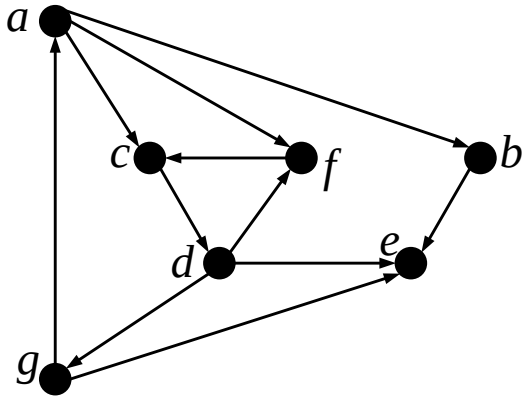
Time Interval

$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

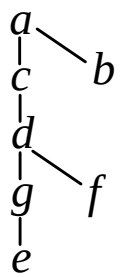
Classify each edge of G into: **forward edge**, **back edge**, and **cross edge**.

Cycle Theorem: Let T be an **arbitrary** DFS-forest. G contains a cycle **if and only if** there is a back edge with respect to T.

Second Step: Find Back Edges



DFS Tree



Time Interval

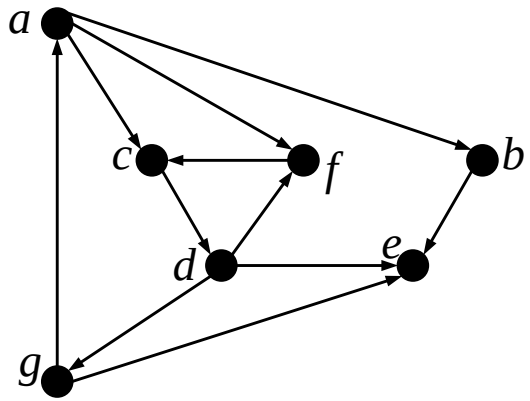
$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

Each edge can be classified in $O(1)$ time using the Parenthesis Theorem.

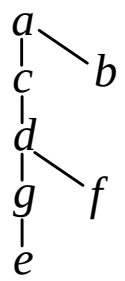
Theorem (the Parenthesis Theorem): Let u and v be two distinct vertices in G. Then:

- $I(u)$ contains $I(v)$ **if and only if** u is an ancestor of v in the DFS-forest.
- $I(u)$ and $I(v)$ are disjoint **if and only if** neither u nor v is an ancestor of the other.

Second Step: Find Back Edges



DFS Tree



Time Interval

$I(a) = [1, 14]$
$I(c) = [2, 11]$
$I(d) = [3, 10]$
$I(g) = [4, 7]$
$I(e) = [5, 6]$
$I(f) = [8, 9]$
$I(b) = [12, 13]$

If G is cyclic, how would you use DFS to find a cycle?

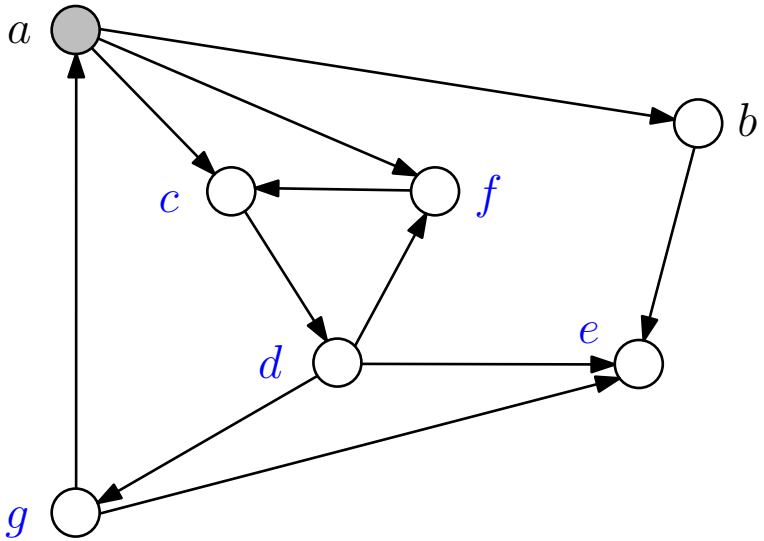
Next, we revisit the **white path theorem**, by far the most important property of DFS.

White Path Theorem: Let u be a vertex in G. Consider the moment right before u enters the stack in the DFS algorithm. Then, a vertex v becomes a proper descendant of u in the DFS-forest **if and only** if the following is true at this moment:

- there is a path from u to v including only white vertices.

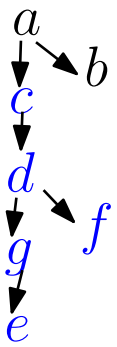
Example 1

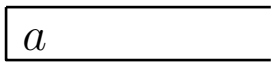
Right before c is pushed into the stack, we have:



DFS Tree

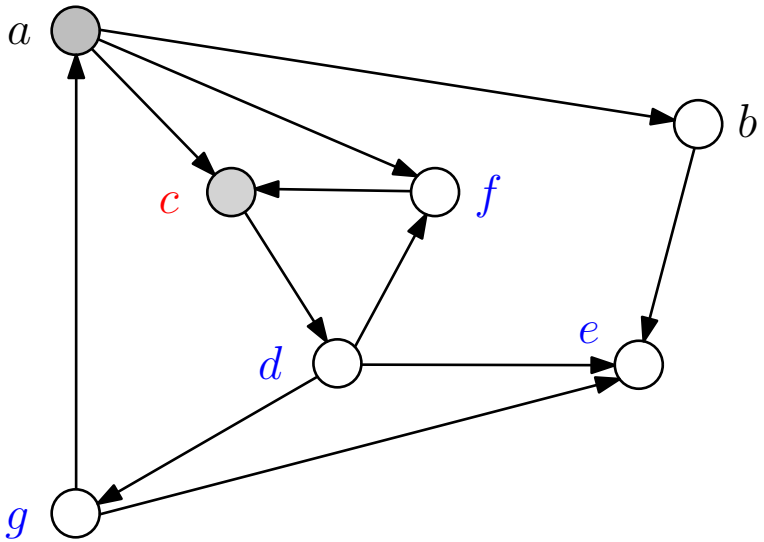
Final DFS Tree



$S =$  $\leftarrow c$

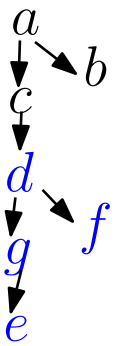
Example 2

Right before d is pushed into the stack, we have:



DFS Tree

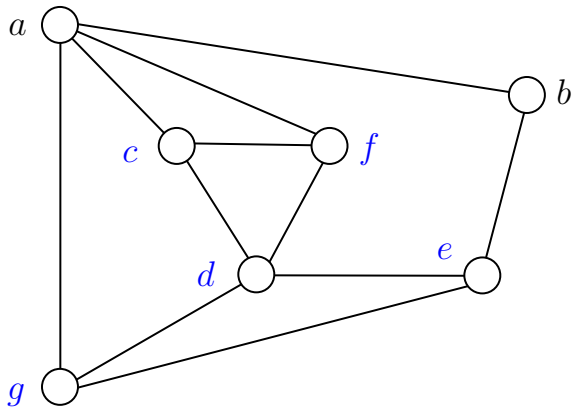
Final DFS Tree



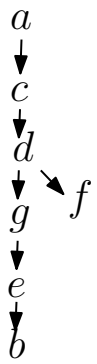
$S =$ $\boxed{ac\quad\quad\quad}$ $\leftarrow d$

So far our discussion has focused on directed graphs. Next, we will see how DFS executes on undirected graphs.

DFS on an Undirected Graph



Final DFS Tree



Classify each edge of G into

- **tree edge** if it appears in the DFS tree produced;
- **back edge** otherwise.

Lemma: For each edge $\{u, v\}$ in G, it must hold that, in the DFS tree produced, either u is an ancestor of v or v is an ancestor of u.

Try proving the lemma with the White Path Theorem.