带边标记的DFS算法

算法实现和正确性

计算机科学与技术系 李松原

Nov. 4th, 2020

边标记

边标记

Tree Edges

Back Edges

Forward Edges

Cross Edges

边标记

Tree Edges

Back Edges

Forward Edges

Cross Edges

First Access!

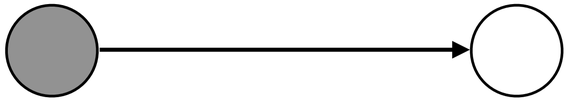
边标记

Tree Edges

Back Edges

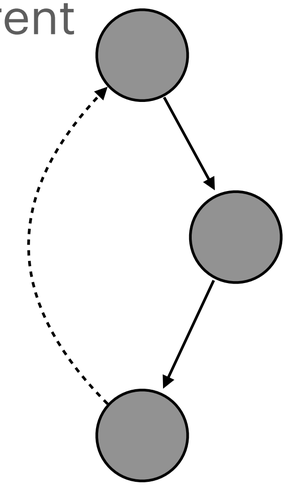
Forward Edges

Cross Edges



First Access!

A Gray Parent



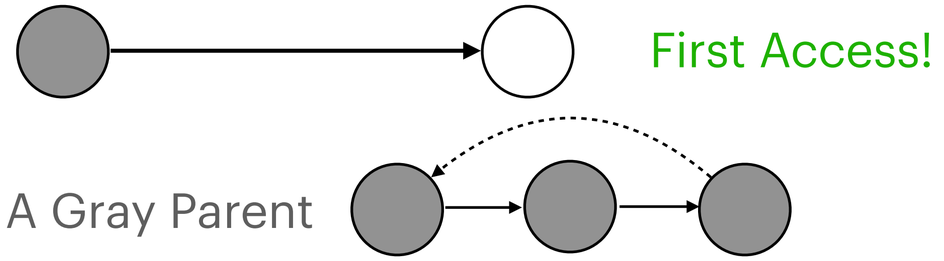
边标记

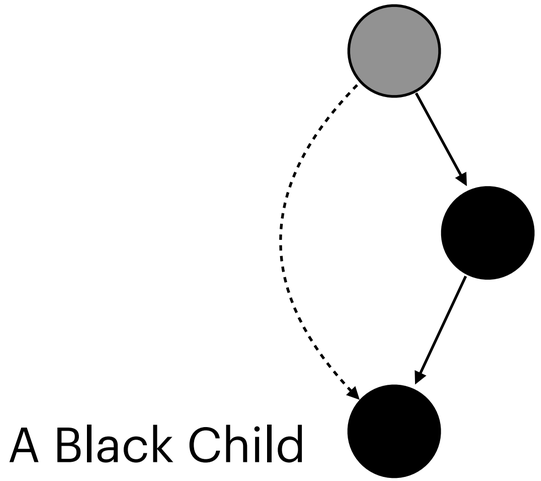
Tree Edges

Back Edges

Forward Edges

Cross Edges





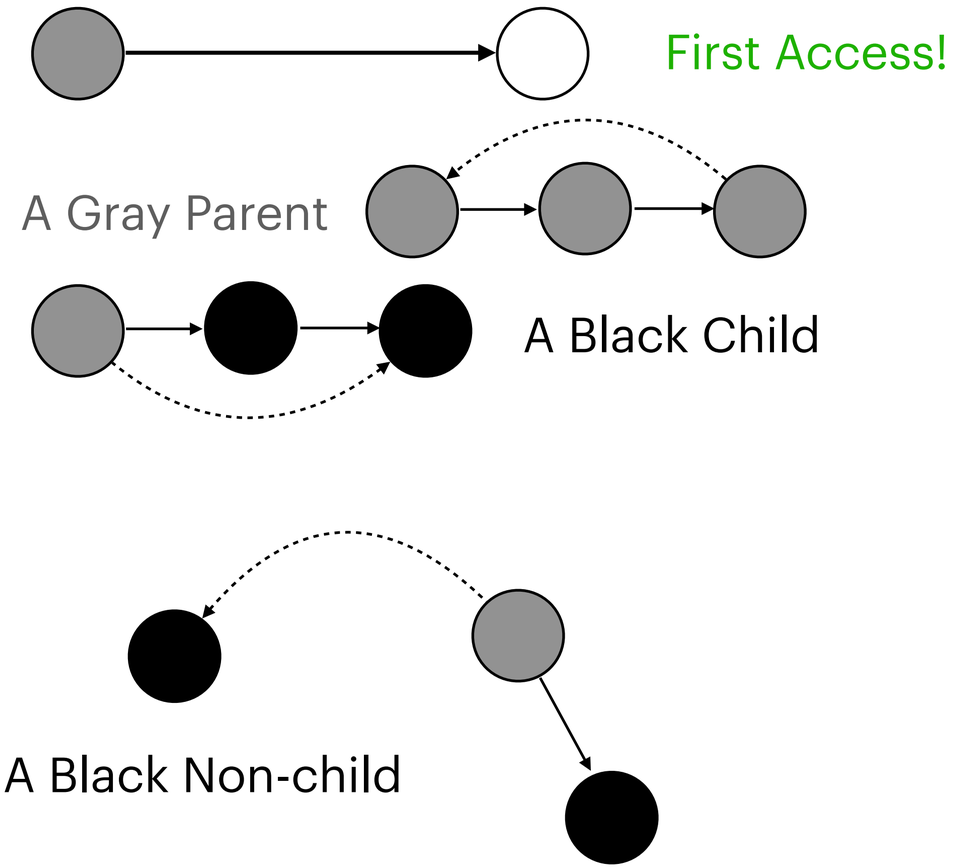
边标记

Tree Edges

Back Edges

Forward Edges

Cross Edges



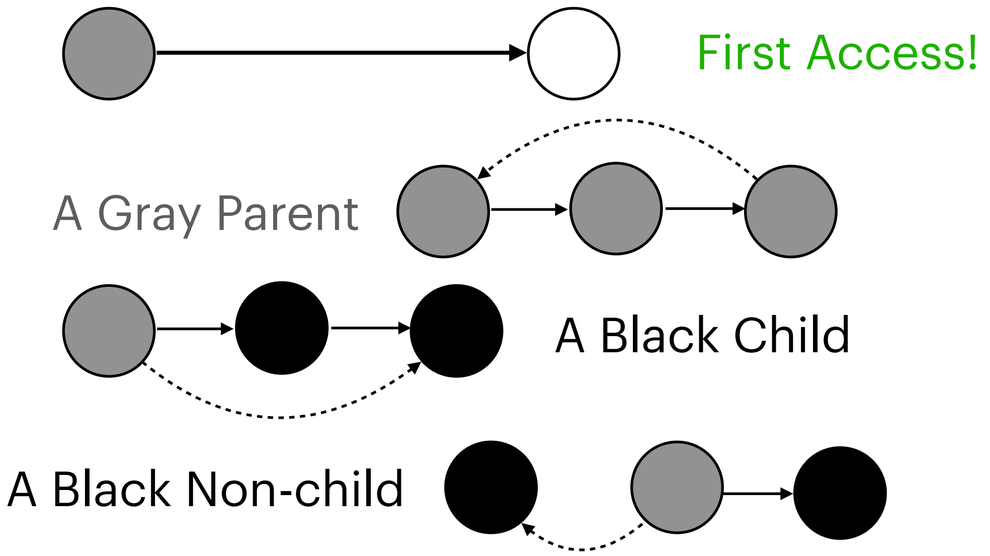
边标记

Tree Edges

Back Edges

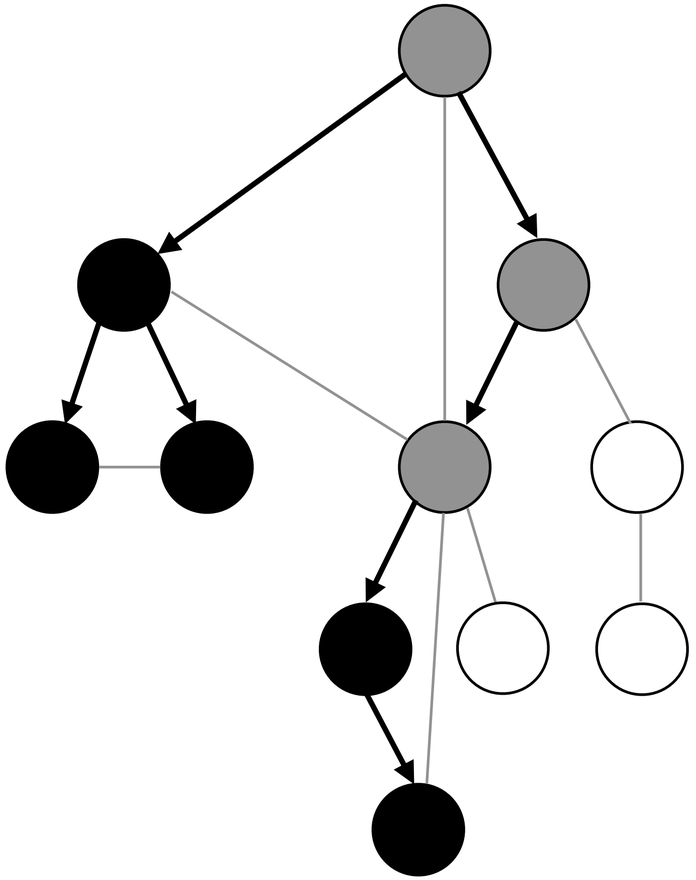
Forward Edges

Cross Edges

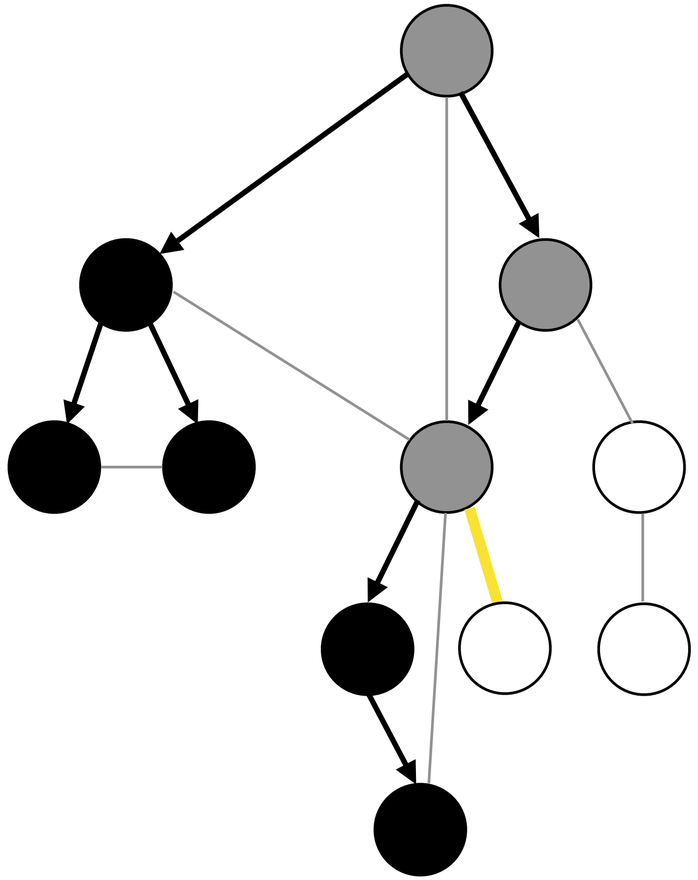


```
procedure DFS-Visit(G, u)
    u.color = GRAY
    for each vertex v adjacent to u
        if v.color == WHITE
            DFS-Visit(G, v)
    u.color = BLACK
```

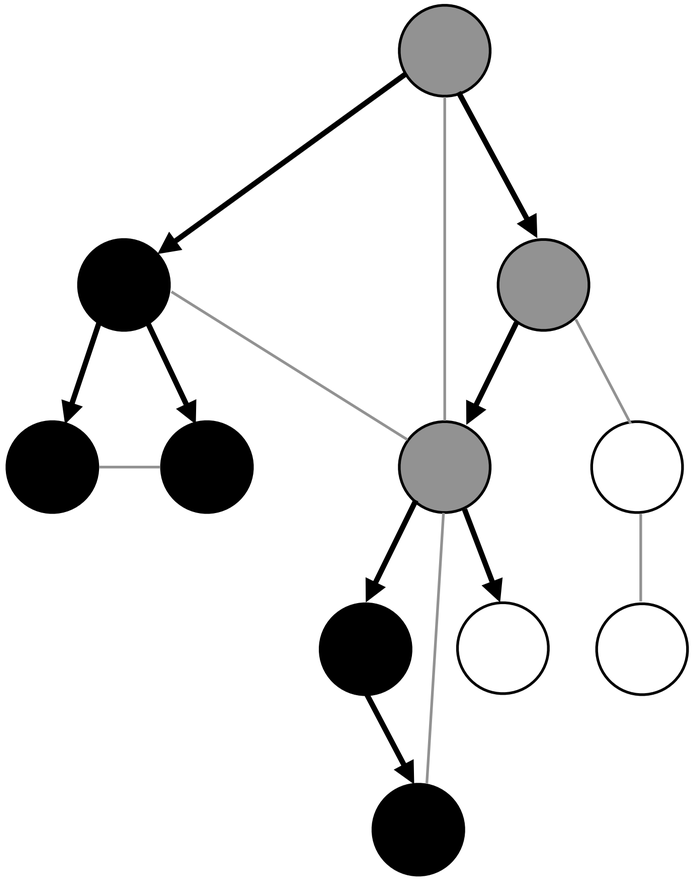
```
procedure DFS-Visit(G, u)
    u.color = GRAY
    for each vertex v adjacent to u
        if v.color == WHITE
            DFS-Visit(G, v)
    u.color = BLACK
```



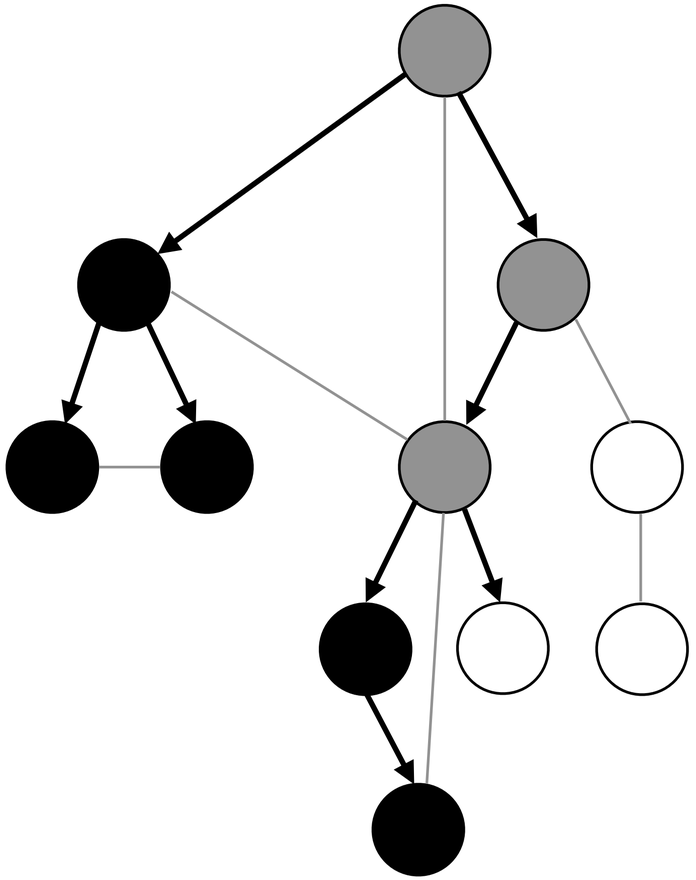
```
procedure DFS-Visit(G, u)
    u.color = GRAY
    for each vertex v adjacent to u
        if v.color == WHITE
            DFS-Visit(G, v)
    u.color = BLACK
```



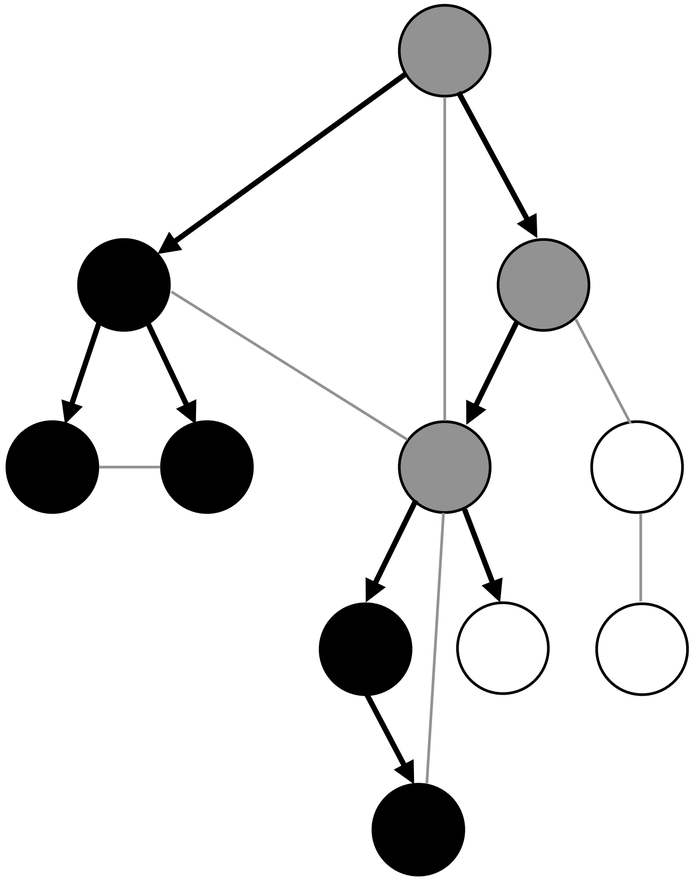
```
procedure DFS-Visit(G, u)
    u.color = GRAY
    for each vertex v adjacent to u
        if v.color == WHITE
            Mark (u, v) as tree edge
            DFS-Visit(G, v)
    u.color = BLACK
```



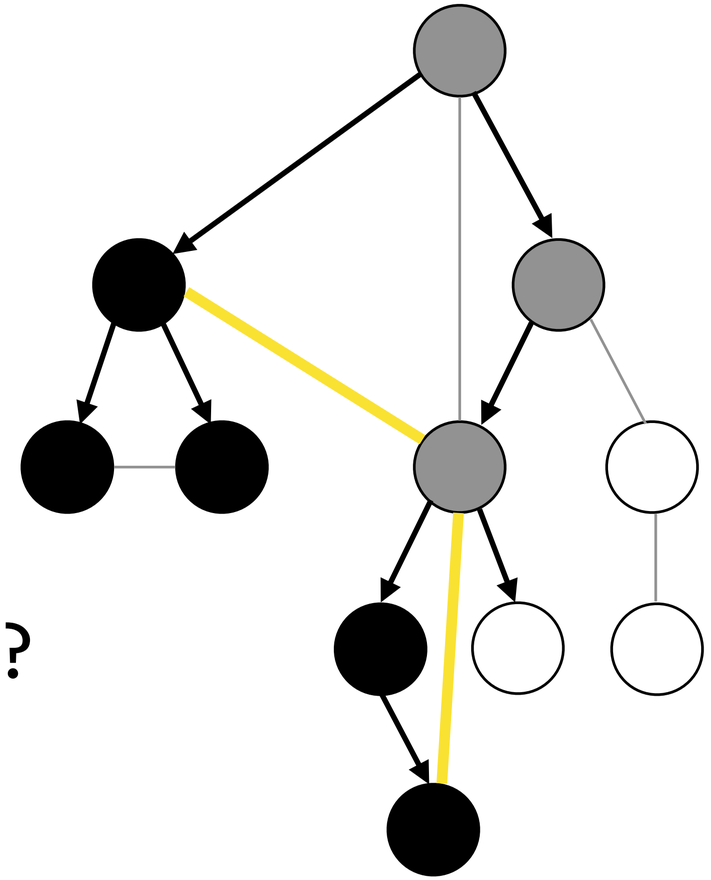
```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
```



```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
else if v.color == GRAY
   Mark (u, v) as back edge
```

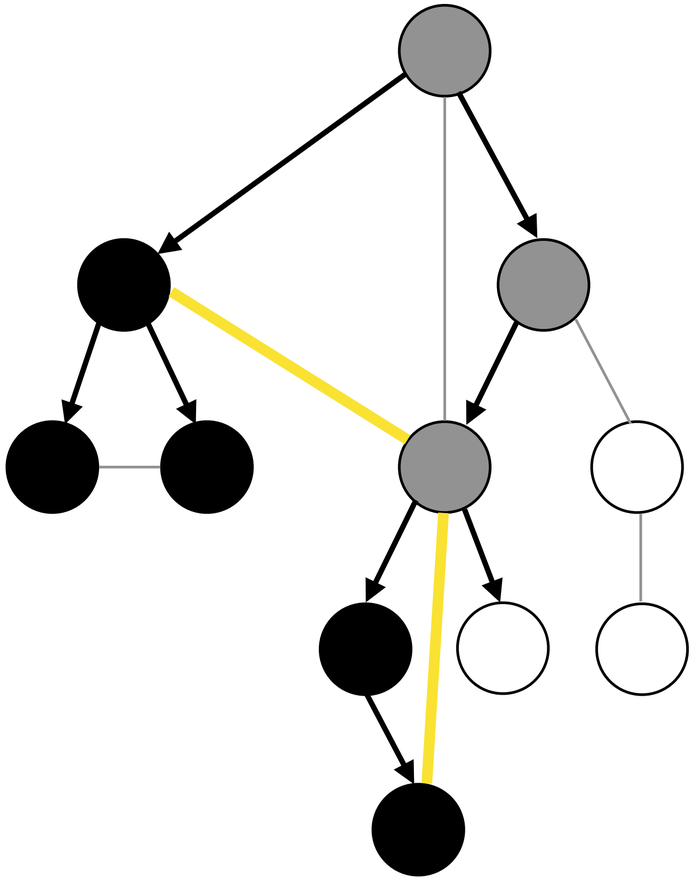


```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
else if v.color == GRAY
   Mark (u, v) as back edge
else if v.color == BLACK
   Mark (u, v) as forward/cross edge?
```

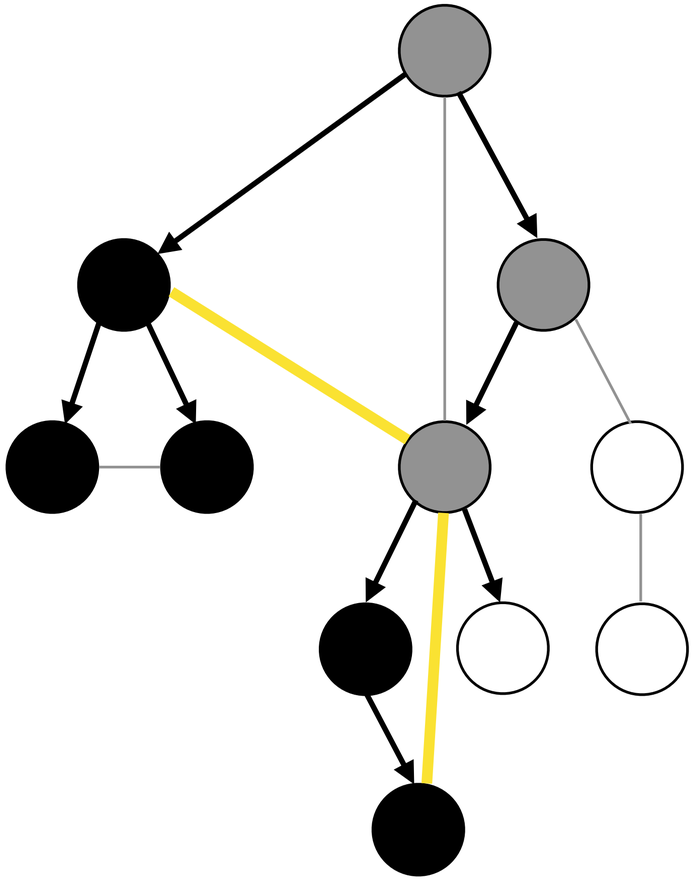


```
else if v.color == BLACK
```

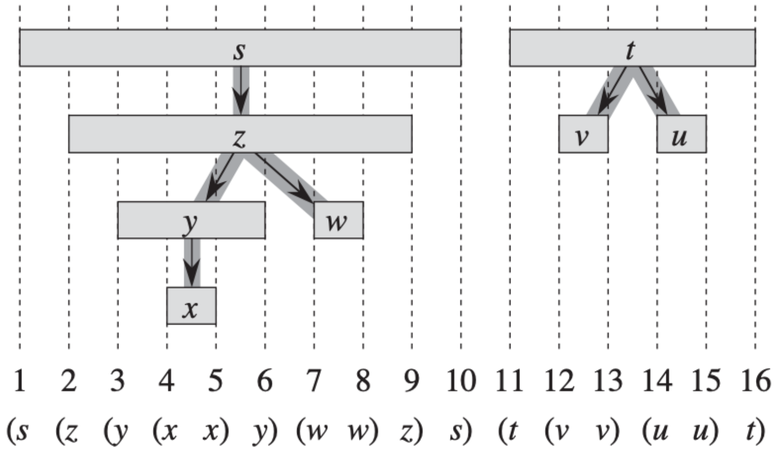
Mark (u, v) as forward/cross edge?

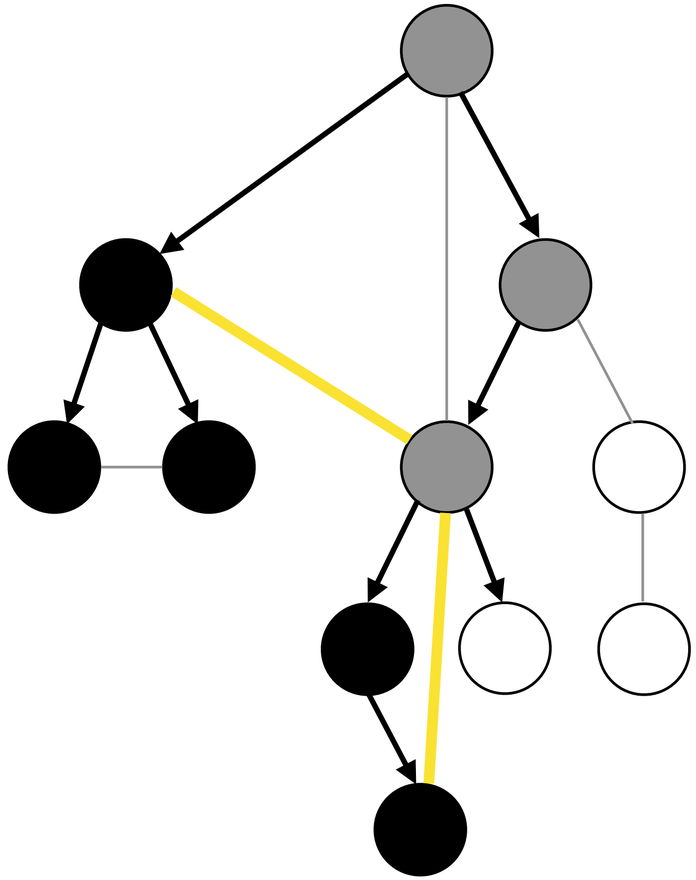


```
else if v.color == BLACK
   if v.f <= u.d
      Mark (u, v) as forward edge
   else
      Mark (u, v) as cross edge
```

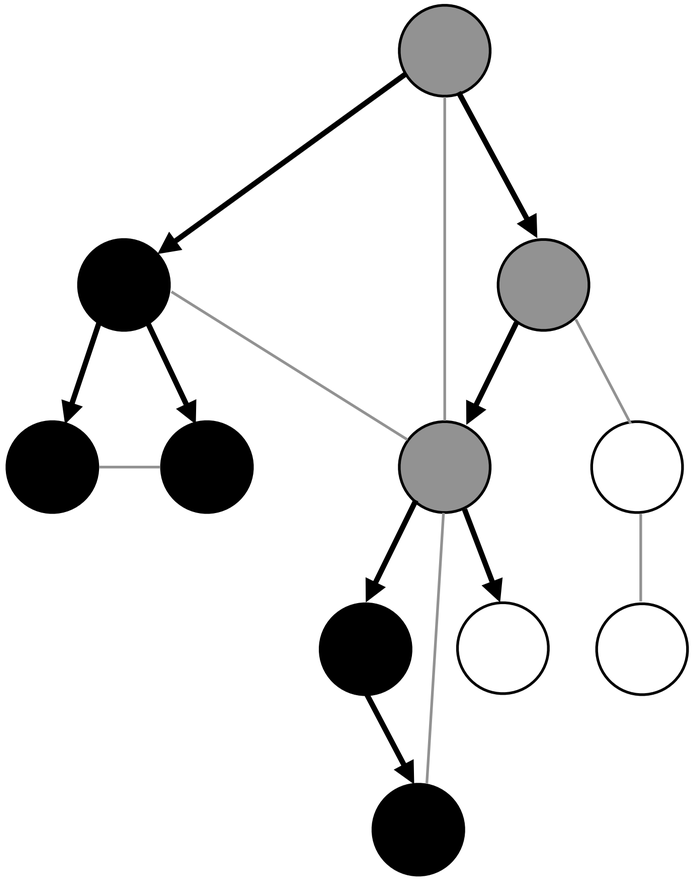


```
else if v.color == BLACK
   if v.f <= u.d
      Mark (u, v) as forward edge
   else
      Mark (u, v) as cross edge
```





```
else if v.color == BLACK
   if v.f <= u.d
      Mark (u, v) as forward edge
   else
      Mark (u, v) as cross edge
```



```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

正确性证明

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

是否所有边都被标记?

```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
else if v.color == GRAY
   Mark (u, v) as back edge
else if v.color == BLACK
   if v.f <= u.d then Mark (u, v) as forward edge
   else  Mark (u, v) as cross edge
```

树边（Tree Edge）

标记的树边是否都为树边?

所有树边是否都被标记?

1. ***Tree edges*** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).

```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
else if v.color == GRAY
   Mark (u, v) as back edge
else if v.color == BLACK
   if v.f <= u.d then Mark (u, v) as forward edge
   else  Mark (u, v) as cross edge
```

1. ***Tree edges*** are edges in the depth-first forest G_π. Edge (u, v) is a tree edge if v was first discovered by exploring edge (u, v).

```
if v.color == WHITE
   Mark (u, v) as tree edge
   DFS-Visit(G, v)
else if v.color == GRAY
   Mark (u, v) as back edge
else if v.color == BLACK
   if v.f <= u.d then Mark (u, v) as forward edge
   else  Mark (u, v) as cross edge
```

后向边（Back Edge）？

2. ***Back edges*** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. We consider self-loops, which may occur in directed graphs, to be back edges.

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

2. ***Back edges*** are those edges (u, v) connecting a vertex u to an ancestor v in a depth-first tree. We consider self-loops, which may occur in directed graphs, to be back edges.

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

前向边（Forward Edge）？

3. ***Forward edges*** are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

3. ***Forward edges*** are those nontree edges (u, v) connecting a vertex u to a descendant v in a depth-first tree.

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

4. ***Cross edges*** are all other edges. They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.

无向图

无向图

Theorem 22.10
In a depth-first search of an undirected graph G, every edge of G is either a tree edge or a back edge.

Proof Let (u, v) be an arbitrary edge of G, and suppose without loss of generality that $u.d < v.d$. Then the search must discover and finish v before it finishes u (while u is gray), since v is on u's adjacency list. If the first time that the search explores edge (u, v), it is in the direction from u to v, then v is undiscovered (white) until that time, for otherwise the search would have explored this edge already in the direction from v to u. Thus, (u, v) becomes a tree edge. If the search explores (u, v) first in the direction from v to u, then (u, v) is a back edge, since u is still gray at the time the edge is first explored. ■

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else if v.color == GRAY
  Mark (u, v) as back edge
else if v.color == BLACK
  if v.f <= u.d then Mark (u, v) as forward edge
  else  Mark (u, v) as cross edge
```

```
if v.color == WHITE
  Mark (u, v) as tree edge
  DFS-Visit(G, v)
else
  Mark (u, v) as back edge
```

感谢聆听

Thanks for your listening!

Enjoy the rest of the week!