Depth First Search

Depth First Search

- Assume unweighted directed graph (allow self loops)

DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. **DFS-Visit(G,u)**

DFS-Visit(G,u)

1. time = time + 1
2. **u.d = time**
3. **u.color = GRAY**
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. **DFS-Visit(G,v)**
8. **u.color = BLACK**
9. time = time + 1
10. **u.f = time**

Each vertex has two timestamps:

$\boldsymbol{u.d}$: First discovered and GRAYed

$\boldsymbol{u.f}$: Finished and Blackened

$\boldsymbol{\forall u: 1 \leq u.d < u.f \leq 2n}$

```
DFS(G)
1.  For each vertex u ∈ G.V
2.      u.color = WHITE
3.      u.π = NIL
4.  time = 0
5.  For each vertex u ∈ G.V
6.      if u.color == WHITE
7.          DFS-Visit(G,u)

DFS-Visit(G,u)
1.  time = time + 1
2.  u.d = time
3.  u.color = GRAY
4.  For each vertex v ∈ G.Adj[u]
5.      if v.color == WHITE
6.          v.π = u
7.          DFS-Visit(G,v)
8.  u.color = BLACK
9.  time = time + 1
10. u.f = time
```

Each vertex has two timestamps:

$u.d$: First discovered and GRAYed

$u.f$: Finished and Blackened

$\forall u: 1 \leq u.d < u.f \leq 2n$

There is no special source s

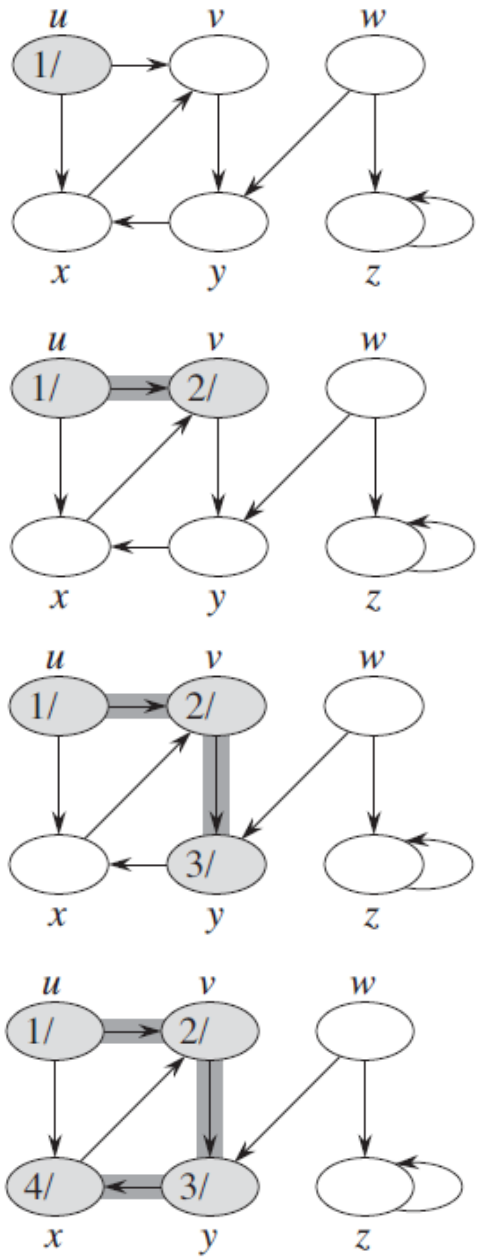
We visit all vertices

DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

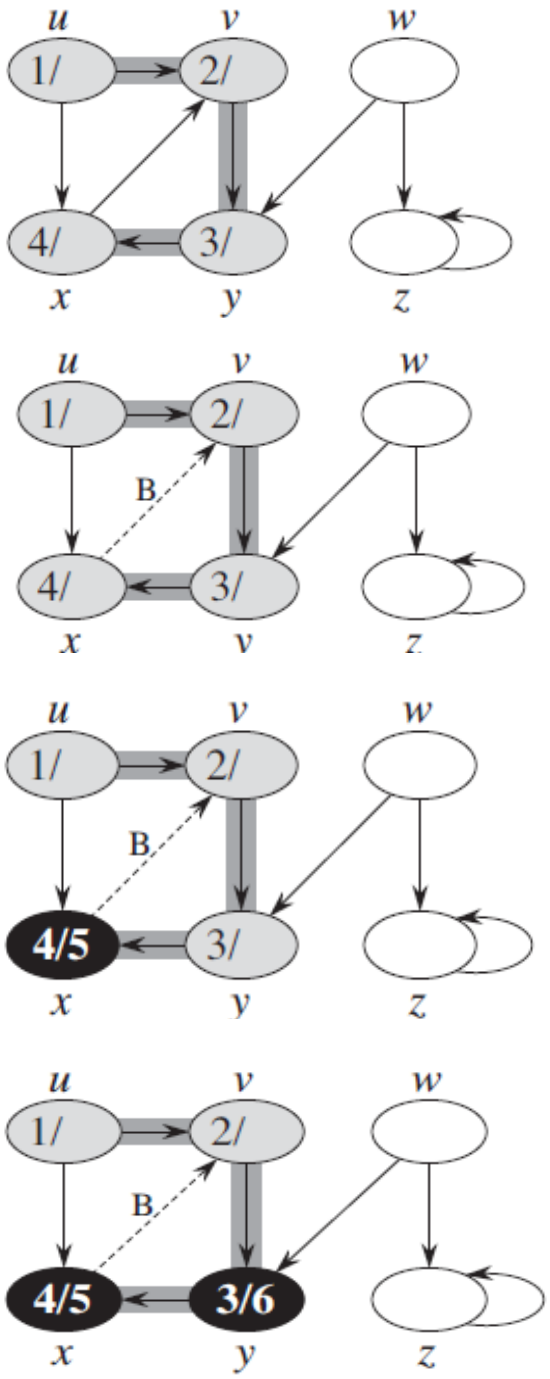


DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

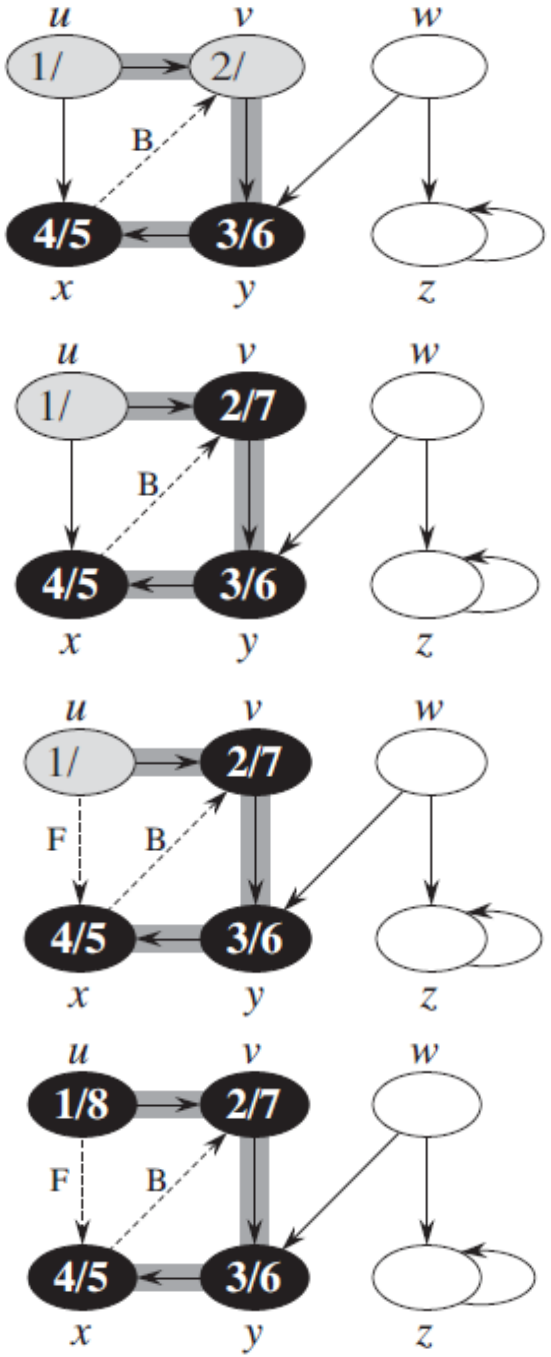


DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

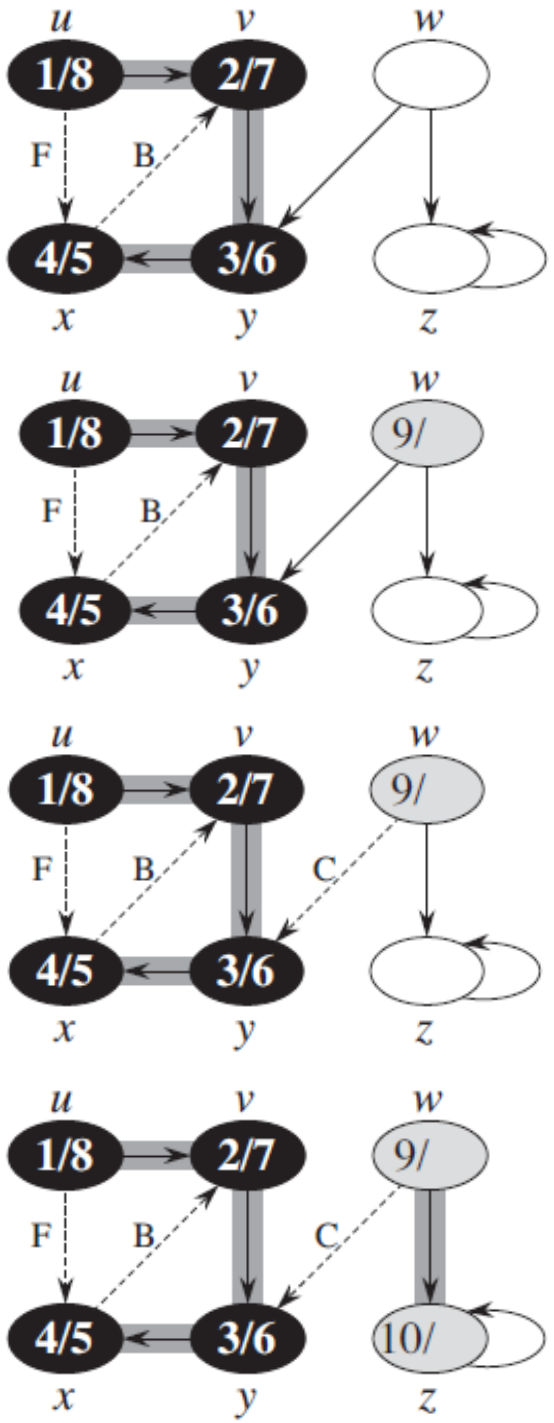


DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

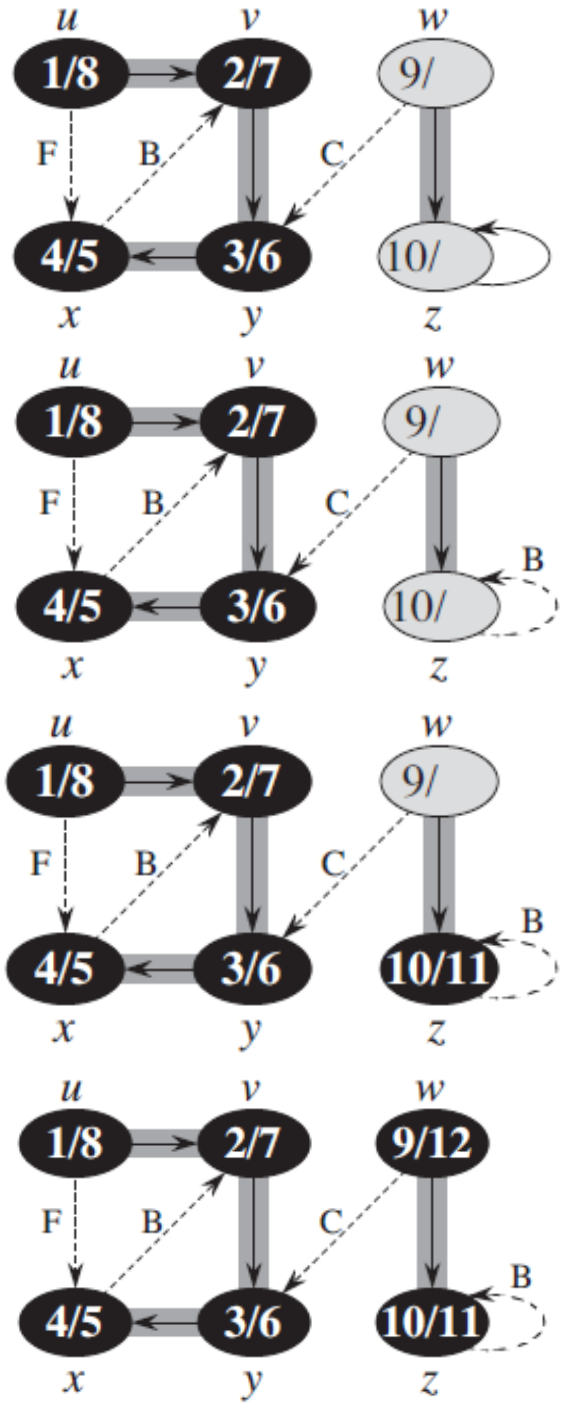


DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

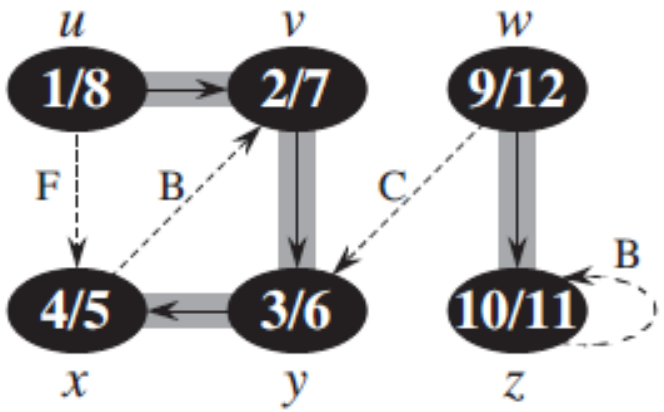


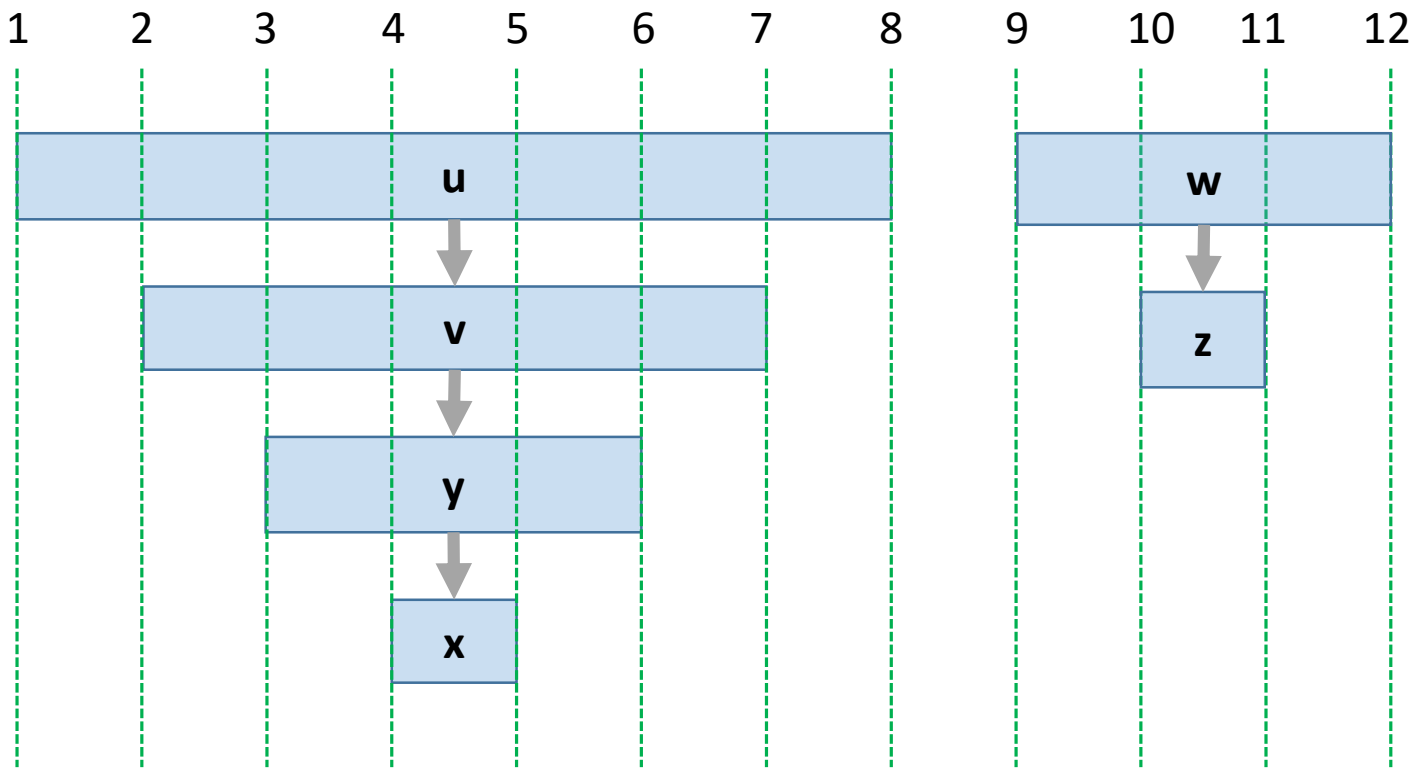
```
DFS(G)
1.  For each vertex u ∈ G.V
2.      u.color = WHITE
3.      u.π = NIL
4.  time = 0
5.  For each vertex u ∈ G.V
6.      if u.color == WHITE
7.          DFS-Visit(G,u)

DFS-Visit(G,u)
1.  time = time + 1
2.  u.d = time
3.  u.color = GRAY
4.  For each vertex v ∈ G.Adj[u]
5.      if v.color == WHITE
6.          v.π = u
7.          DFS-Visit(G,v)
8.  u.color = BLACK
9.  time = time + 1
10. u.f = time
```

Lem 1: The graph $G_\pi = (V, E_\pi)$, $E_\pi = \{ (v_\pi, v) \mid v_\pi \neq NIL \}$ is a forest (DFS-forest)

Proof: It corresponds to the structure of the recursive calls to DFS-Visit □


u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z


1
2
3
4
5
6
7
8
9
10
11
12
u
v
y
x
w
z

DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. $u.\pi$ = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

DFS-Visit(G,u)

1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. $v.\pi$ = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

Thm 2: $[v.d, v.f] \subset [u.d, u.f]$ iff v is a descendant of u in the DFS-forest, $[v.d, v.f] \cap [u.d, u.f] = \emptyset$ iff u and v are unrelated in the DFS-forest

Proof: Since the interval of v is the time in which the recursive call DFS-Visit(G,v) is active.

□

DFS(G)

1. For each vertex $u \in G.V$
2. u.color = WHITE
3. u.π = NIL
4. time = 0
5. For each vertex $u \in G.V$
6. if u.color == WHITE
7. DFS-Visit(G,u)

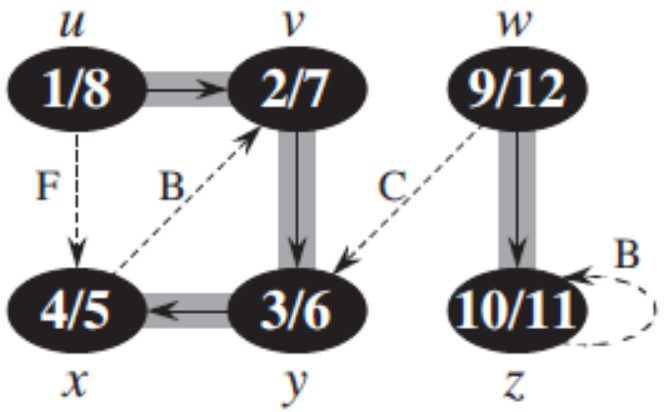
DFS-Visit(G,u)

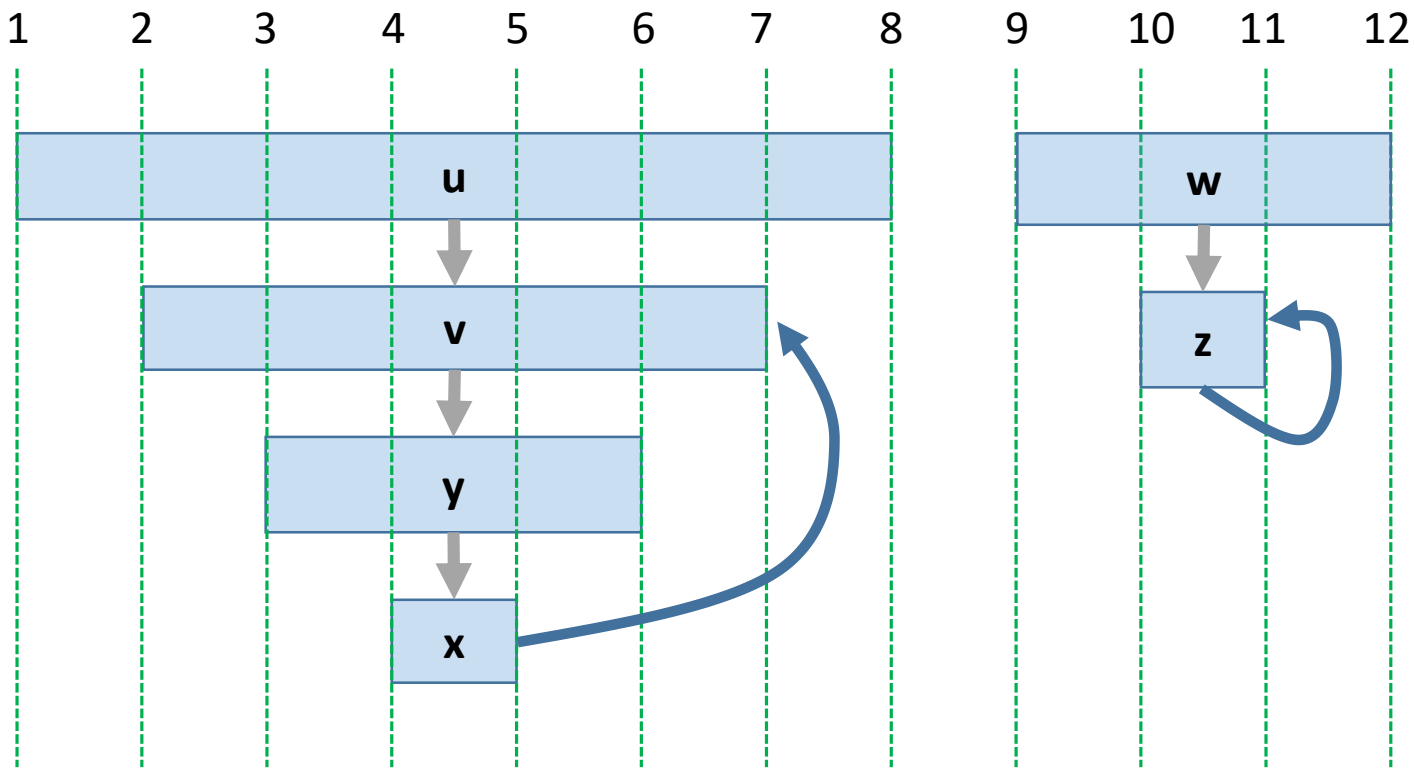
1. time = time + 1
2. u.d = time
3. u.color = GRAY
4. For each vertex $v \in G.Adj[u]$
5. if v.color == WHITE
6. v.π = u
7. DFS-Visit(G,v)
8. u.color = BLACK
9. time = time + 1
10. u.f = time

Thm 3: v is a descendant of u in the DFS-forest iff when we invoke DFS-Visit(u) there is a white path from u to v

Edge classification

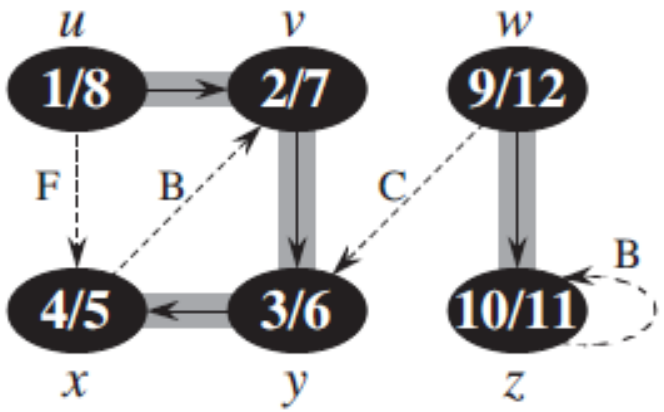
- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest

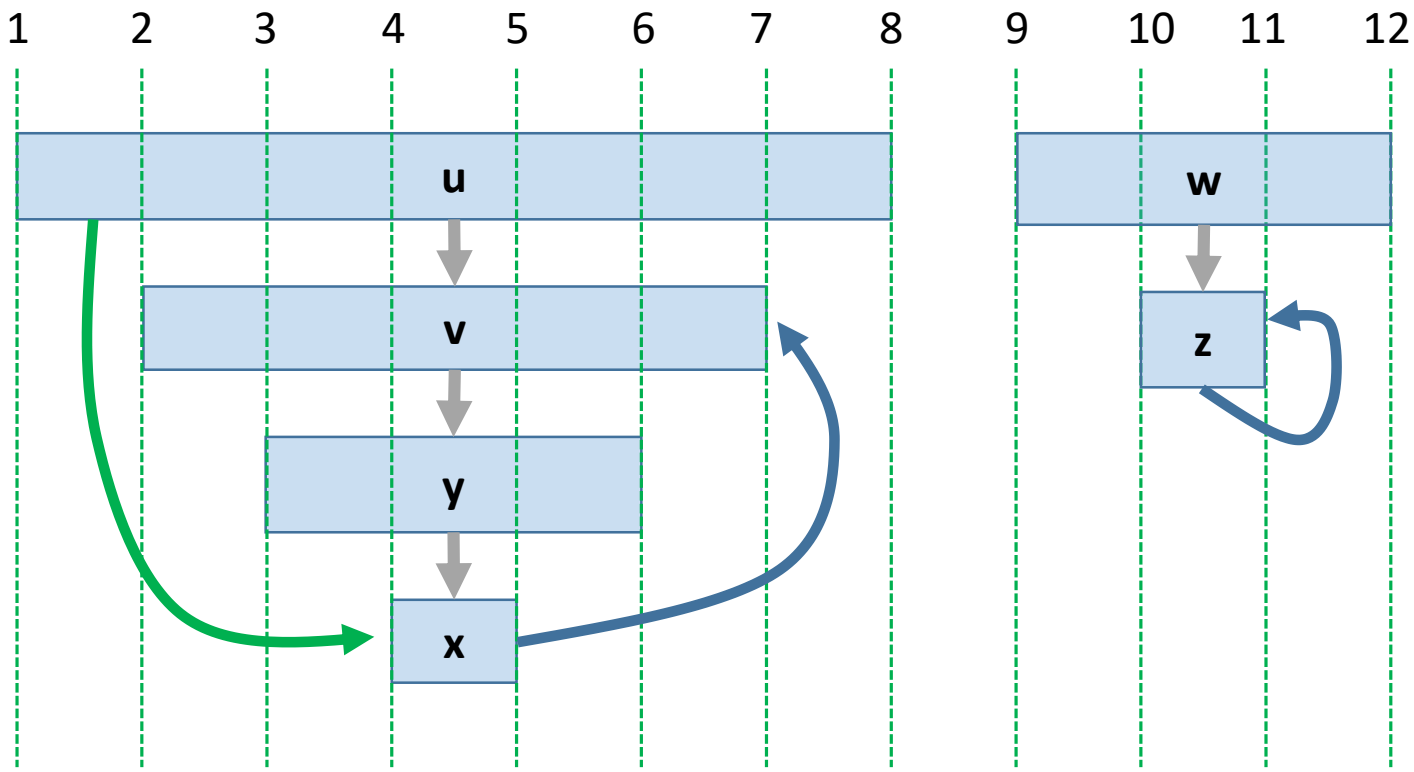

u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z


1
2
3
4
5
6
7
8
9
10
11
12
u
v
y
x
w
z

Edge classification

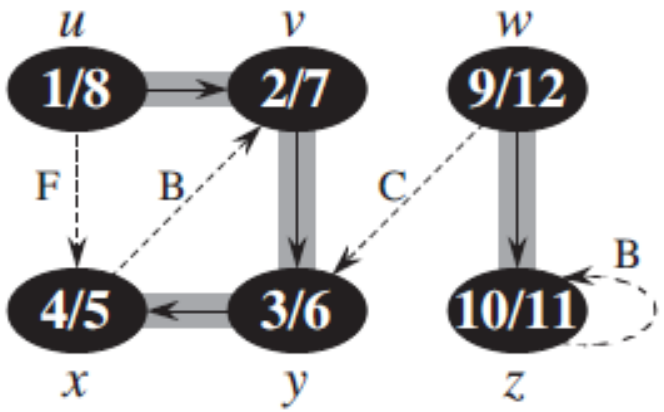
- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest
- Forward edges: nontree edges (u,v) such that v is a descendant of u

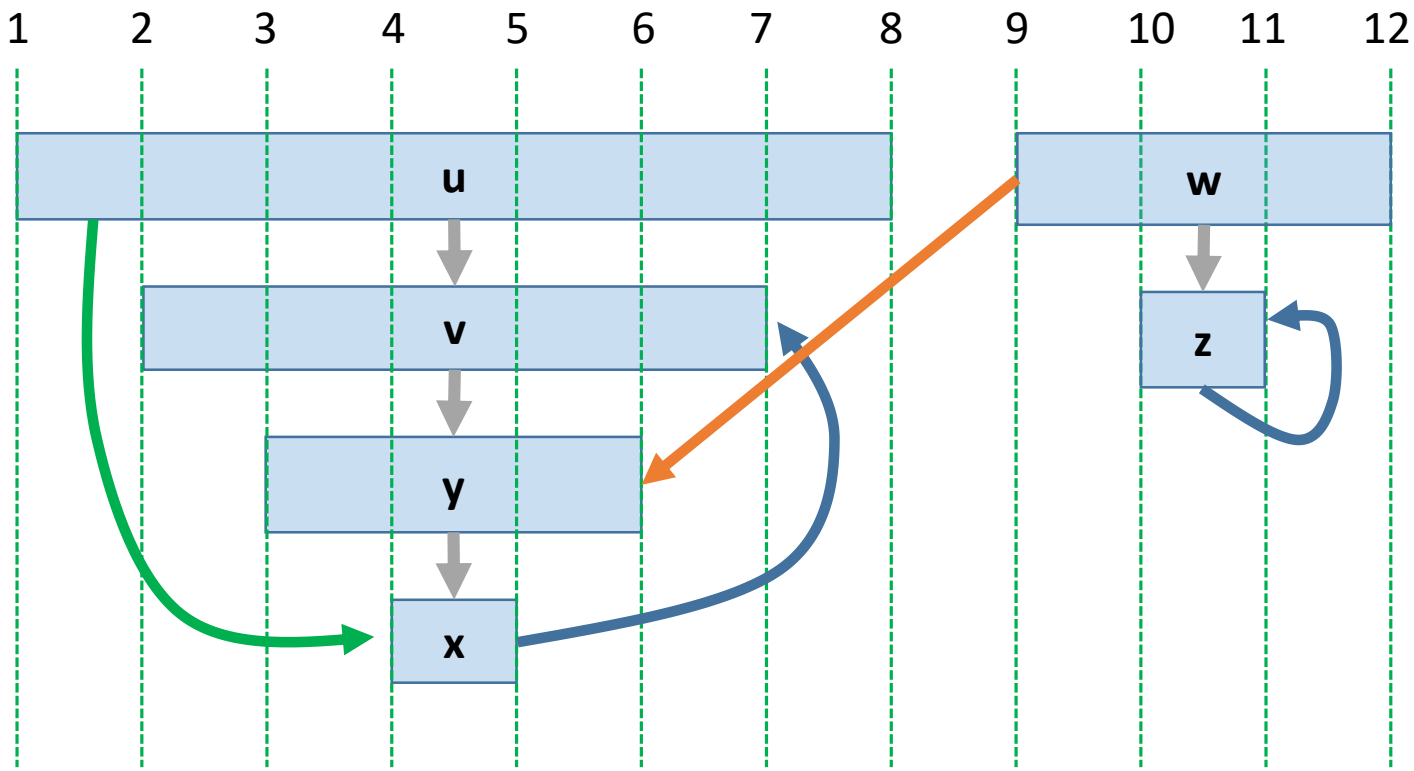

u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z


1
2
3
4
5
6
7
8
9
10
11
12
u
v
y
x
w
z

Edge classification

- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest
- Forward edges: nontree edges (u,v) such that v is a descendant of u
- Cross edges: all other edges


u
v
w
1/8
2/7
9/12
F
B
C
B
4/5
3/6
10/11
x
y
z


1
2
3
4
5
6
7
8
9
10
11
12
u
v
y
x
w
z

Edge classification

- Tree edges: (u,v), DFS-Visit(v) was called from DFS-visit(u)
- Back edges: (u,v) such that v is an ancestor of u in the DFS-forest
- Forward edges: nontree edges (u,v) such that v is a descendant of u
- Cross edges: all other edges

Cross edges

Thm 4: If (u,v) is a cross edge then $u.d > v.f$

Classify as we go..

- Tree edges: (u,v), v is white
- Back edges: (u,v) v is gray $v.d < u.d$
- Forward edges: v is black $v.d > u.d$
- Cross edges: v is black, $v.d < v.f < u.d$