Depth First Search (DFS)

Subhabrata Samajder

IIIT, Delhi

Winter Semester,
20th May, 2023

Depth-first Search (DFS)

Tree and Forest

Definition (Tree)

A connected acyclic graph $G = (V, E)$ is called a **tree**.

Note: For a tree $|E| = |V| - 1$.

Tree and Forest

Definition (Tree)

A connected acyclic graph $G = (V, E)$ is called a **tree**.

Note: For a tree $|E| = |V| - 1$.

Definition (Forest)

A collections of trees is called a **forest**.

Introduction

- **Strategy:** Search "deeper" in the graph whenever possible.
- Explore the unexplored edges leaving the most recently discovered vertex v, first.
- When all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered.
- Continues the process until all the vertices that are reachable from the source vertex s are discovered.
- If undiscovered vertices remain, then
  - select any one of them as the new source.
  - Repeat with the new source until all vertices get discovered.
    - **Note:** This is similar to BFS.

Introduction

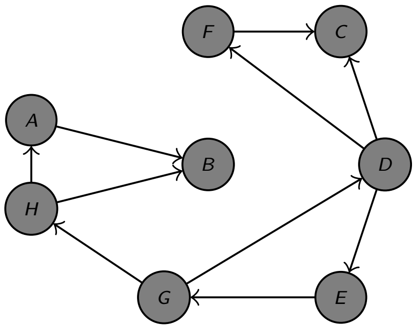
- Similarities with BFS.
  - Whenever a vertex v is discovered during a scan of $Adj[u]$, DFS records this event by setting $\pi[v] = u$.
  - The predecessor subgraph of a DFS is defined as:

    $$G_\pi = (V, E_\pi), \text{ where } E_\pi = \{(\pi[v], v) : v \in V \wedge \pi[v] \neq \text{NIL}\}.$$

  - G_π forms a depth-first forest.
  - The edges in E_π are called tree edges.
  - Works for both directed and undirected graphs.

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.

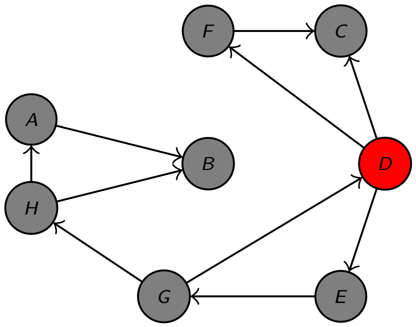


Visited Array

A	
B	
C	
D	
E	
F	
G	
H	

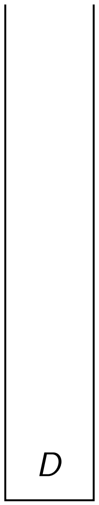
A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	
B	
C	
D	✓
E	
F	
G	
H	

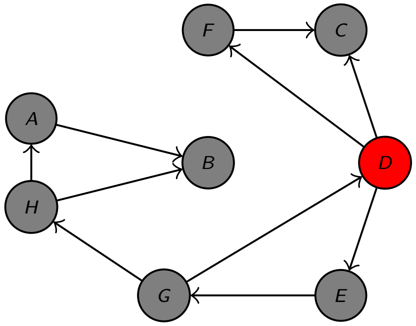


Visit *D*.

The order of visited nodes: *D*

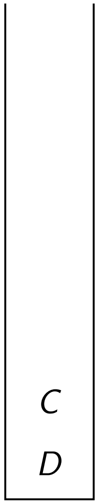
A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	
D	✓
E	
F	
G	
H	



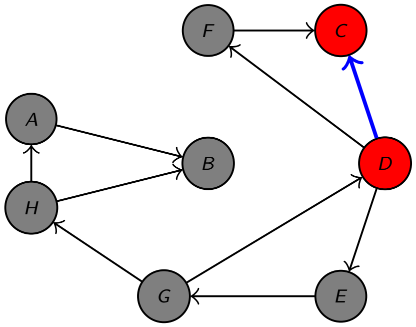
Consider nodes adjacent to D, decide to visit C first.

Rule: Visit adjacent nodes in alphabetical order.

The order of visited nodes: D

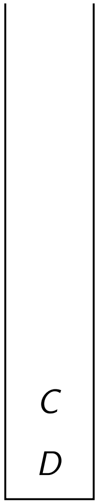
A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	✓
D	✓
E	
F	
G	
H	

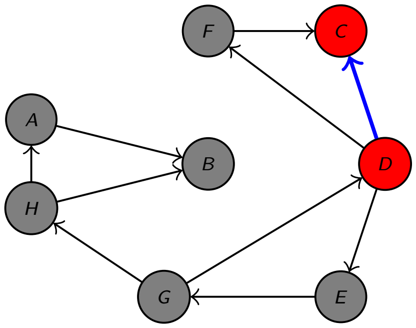


Visit C.

The order of visited nodes: D, C

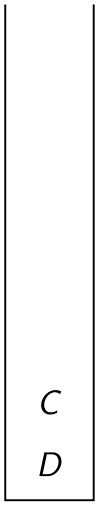
A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	
B	
C	✓
D	✓
E	
F	
G	
H	



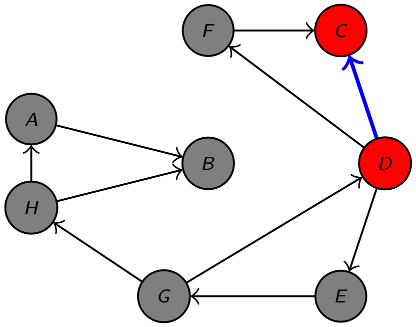
No nodes adjacent to C.

Cannot continue ⇒ backtrack!

The order of visited nodes: *D*, *C*

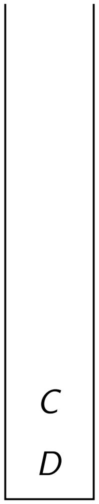
A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	
B	
C	✓
D	✓
E	
F	
G	
H	

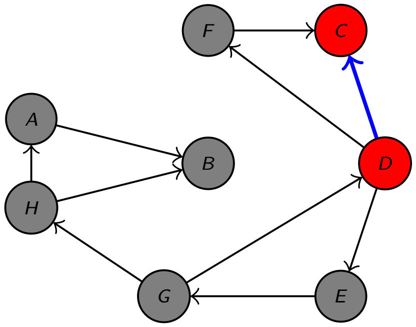


But how?

The order of visited nodes: *D*, *C*

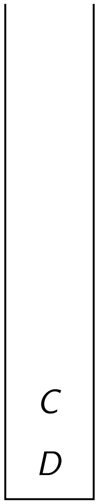
A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	✓
D	✓
E	
F	
G	
H	



C

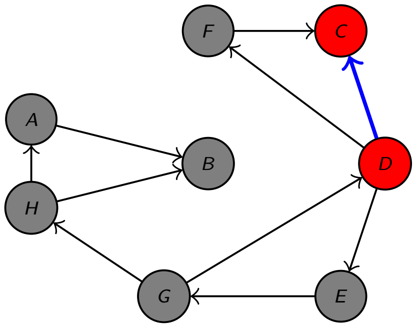
D

Note: Requires a data structure that remembers the sequence of the vertices visited and is able to *delete the last visited vertex efficiently*.

The order of visited nodes: D, C

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	✓
D	✓
E	
F	
G	
H	

Stack

C

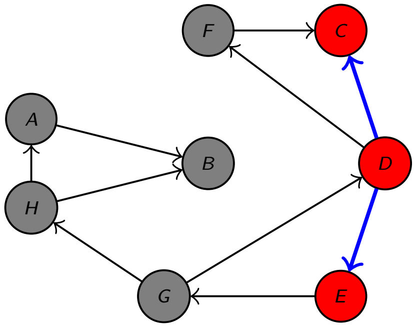
D

Note: Requires a data structure that remembers the sequence of the vertices visited and is able to *delete the last visited vertex efficiently*. Use a stack!

The order of visited nodes: D, C

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	
B	
C	✓
D	✓
E	✓
F	
G	
H	

Stack

E

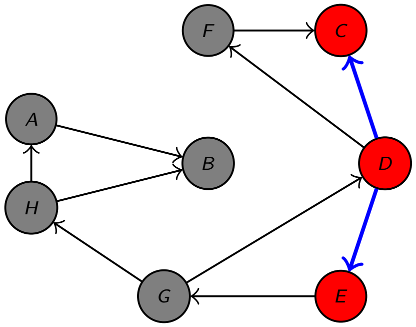
D

Back to *D*; *C* already visited; decide to visit *E* next.

The order of visited nodes: D, C, E

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	✓
D	✓
E	✓
F	
G	
H	

Stack

E

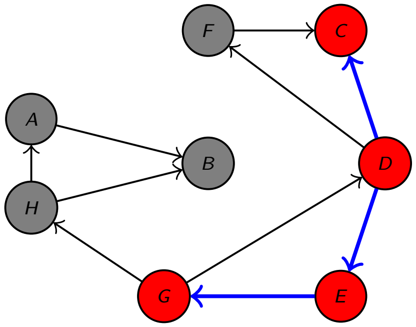
D

Only G is adjacent to E.

The order of visited nodes: D, C, E

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	
B	
C	✓
D	✓
E	✓
F	
G	✓
H	

Stack

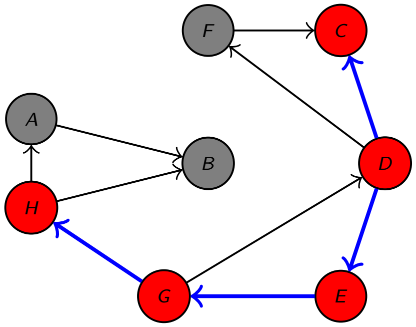
G
E
D

Visit *G*.

The order of visited nodes: *D*, *C*, *E*, *G*

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	
B	
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

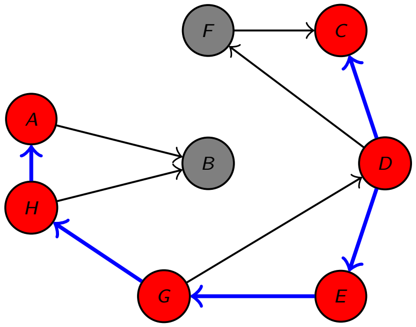
H
G
E
D

Nodes D and H are adjacent to G.
D has already been visited.
Decide to visit H.

The order of visited nodes: D, C, E, G, H

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

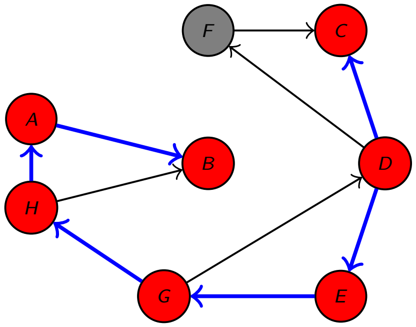
A
H
G
E
D

Nodes A and B are adjacent to H.
Decide to visit A next.

The order of visited nodes: D, C, E, G, H, A

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

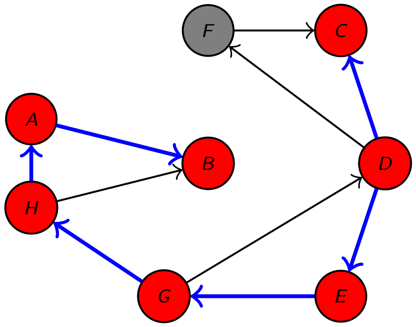
B
A
H
G
E
D

Only Node *B* is adjacent to *A*.
Decide to visit *B* next.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

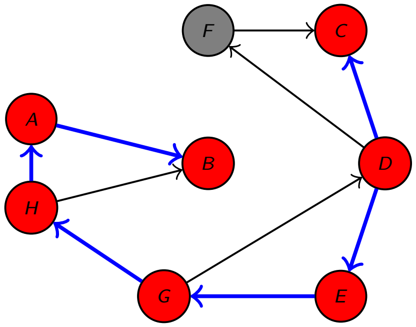
A
H
G
E
D

No unvisited nodes adjacent to B.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

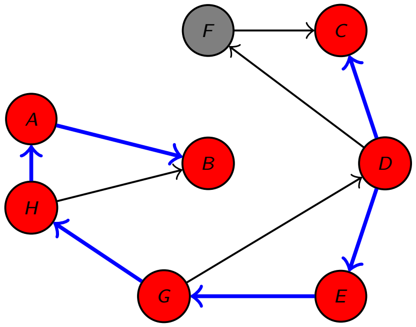
H
G
E
D

No unvisited nodes adjacent to A.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

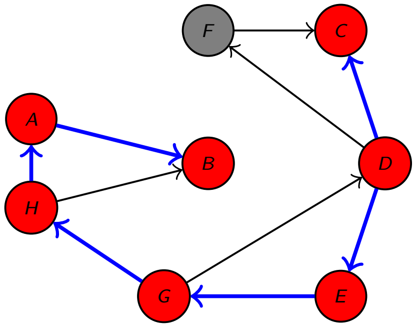
G
E
D

No unvisited nodes adjacent to *H*.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

E

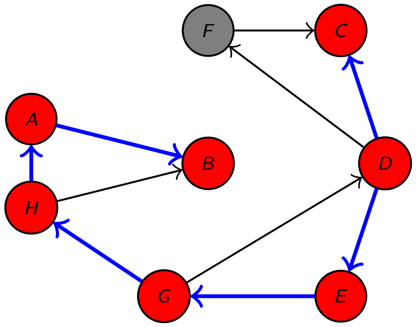
D

No unvisited nodes adjacent to G.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	
G	✓
H	✓

Stack

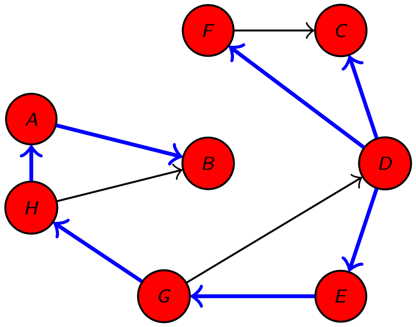
D

No unvisited nodes adjacent to *E*.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

Stack

F

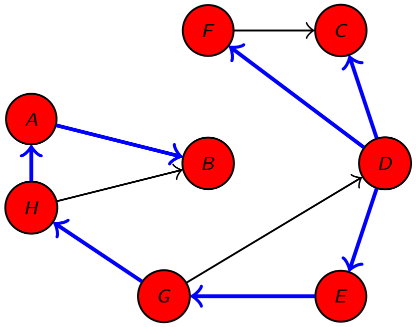
D

Visit F.

The order of visited nodes: D, C, E, G, H, A, B, F

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

Stack

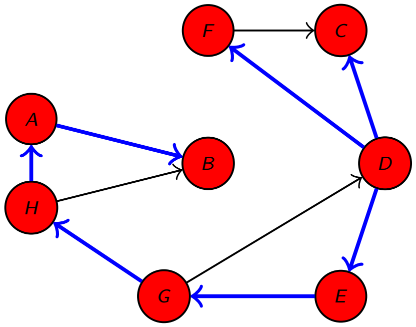
D

No unvisited nodes adjacent to F.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B, F

A Walk Through

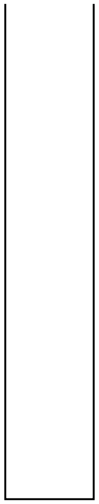
Task: Conduct a DFS of the graph starting with node *D*.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

Stack

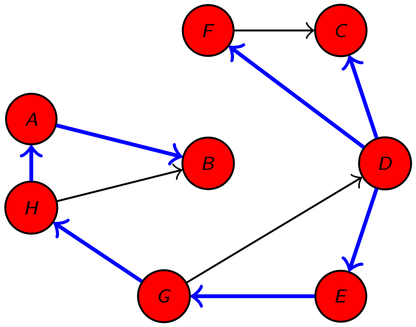


No unvisited nodes adjacent to *D*.
Backtrack - Pop from the stack.

The order of visited nodes: D, C, E, G, H, A, B, F

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

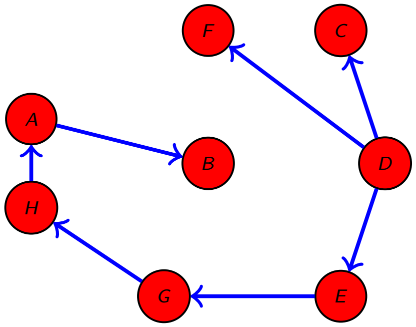
Stack

Stack is **empty** and **no other vertex is left unvisited**.
Depth-first traversal is **done**.

The order of visited nodes: D, C, E, G, H, A, B, F

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

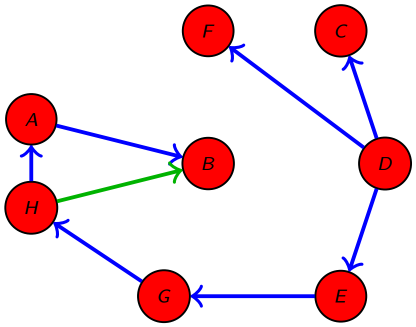
Stack

DFS Tree

The order of visited nodes: D, C, E, G, H, A, B, F

A Walk Through

Task: Conduct a DFS of the graph starting with node D.



Visited Array

A	✓
B	✓
C	✓
D	✓
E	✓
F	✓
G	✓
H	✓

Stack

Note: Does not give the shortest distance from the source vertex.

Example: Shortest distance from D to B is $D \rightarrow E \rightarrow G \rightarrow H \rightarrow B$ instead of $D \rightarrow E \rightarrow G \rightarrow H \rightarrow A \rightarrow B$ given by the DFS tree.

Colouring Scheme

Like BFS, vertices are colored to indicate their state.

- White: Each vertex is initially white.
- Gray: A vertex is coloured gray when it is discovered.
- Black: A vertex is coloured black when all vertices of its adjacency list has been discovered.

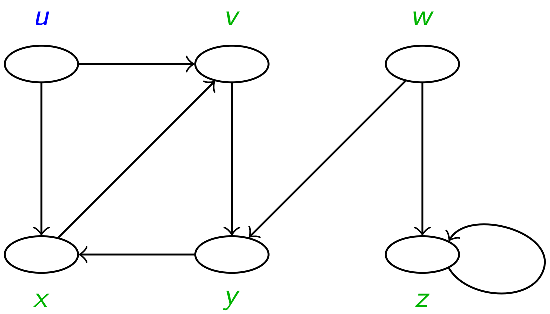
Timestamps

- DFS timestamps each vertex.
- Each vertex v has two timestamps:
  - **Discovery time** $d[v]$**:** v was first discovered (gray).
  - **Finishing time** $f[v]$**:** Finished examining $Adj[v]$ (black).
- Timestamps are integers between 1 and $2|V|$.

Timestamps

- DFS timestamps each vertex.
- Each vertex v has two timestamps:
  - **Discovery time $d[v]$:** v was first discovered (gray).
  - **Finishing time $f[v]$:** Finished examining $Adj[v]$ (black).
- Timestamps are integers between 1 and $2|V|$.
- **Note:** For every vertex u, $d[u] < f[u]$.
- A vertex is therefore
  - WHITE before time $d[u]$,
  - GRAY between time $d[u]$ and time $f[u]$, and
  - BLACK thereafter.
- A global variable *time* is used for timestamping.
- This idea is used in many graph algorithms and are generally helpful in reasoning about the behavior of depth-first search.

DFS(G)

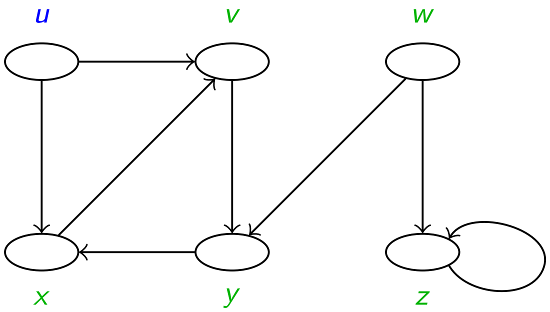


DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    for each vertex $u \in V$
        $color[u] \leftarrow$ WHITE;
        $\pi[u] \leftarrow$ NIL;
```

DFS(G)

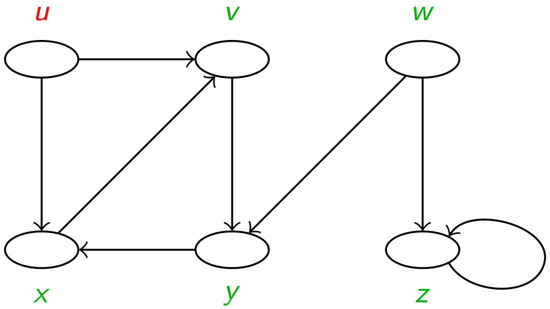


DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

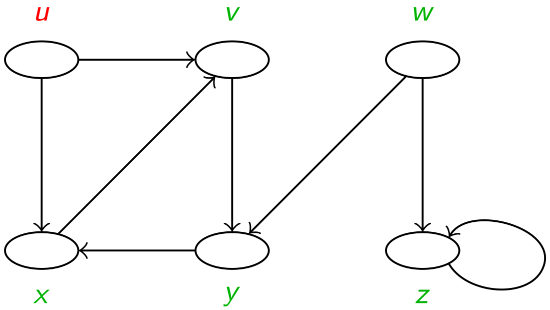
```
Begin
1    for each vertex $u \in V$
2        $color[u] \leftarrow$ WHITE;
3        $\pi[u] \leftarrow$ NIL;
4    $time \leftarrow 0$;
```

DFS(G)



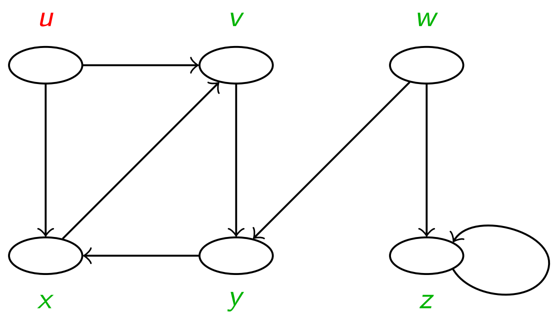
```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
5   for each vertex $u \in V$
```

DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] =$ WHITE)
```

DFS(G)

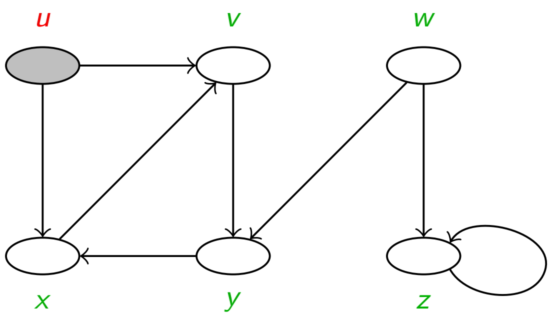


DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

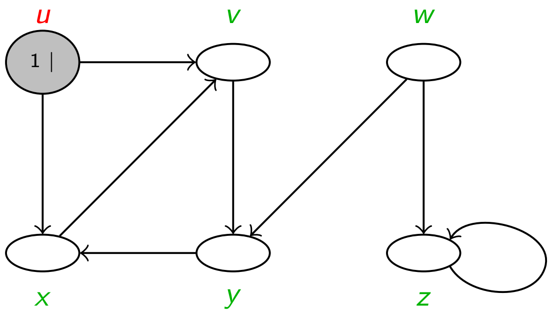
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
```

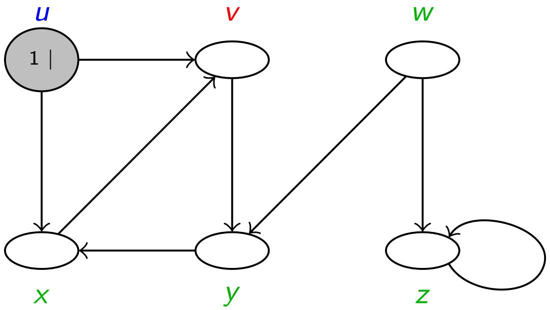
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
      ...
   for each vertex $u \in V$
      if ($color[u] = \text{WHITE}$)
         DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
   $color[u] \leftarrow \text{GRAY}$;      // $u$ discovered.
   $time \leftarrow time + 1$;
   $d[u] \leftarrow time$;
```

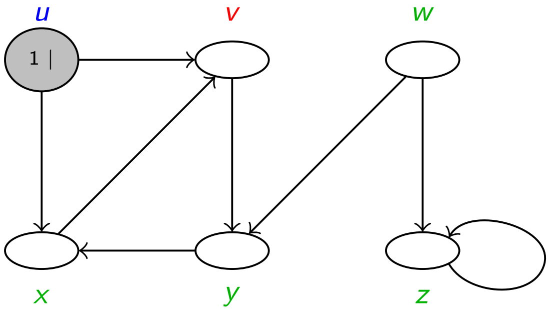
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] =$ WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
```

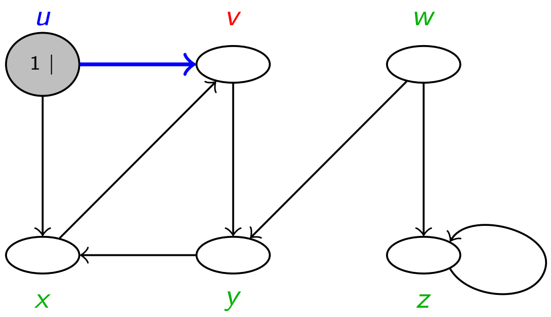
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
```

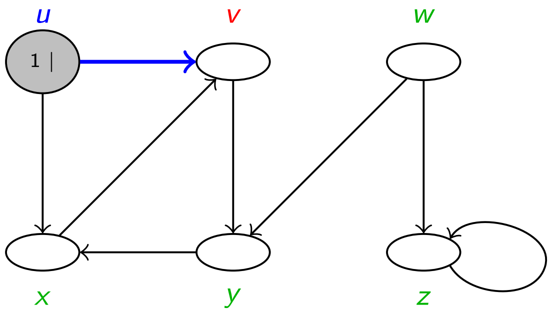
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
```

DFS(G)



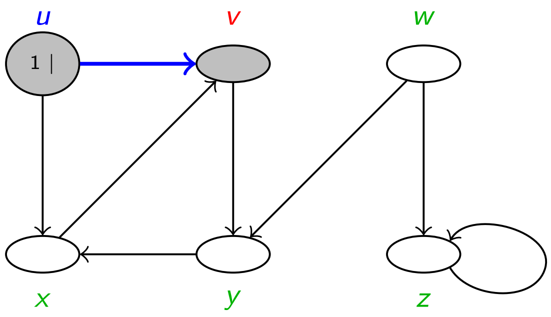
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



DFS(G)

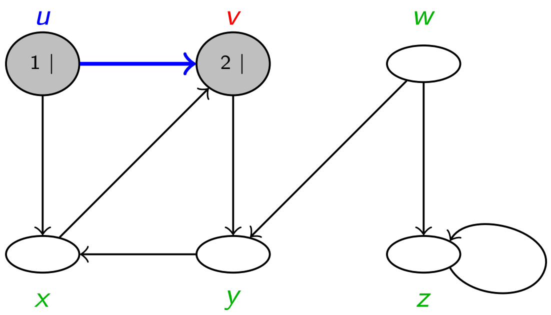
I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

DFS-VISIT(u)

```
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

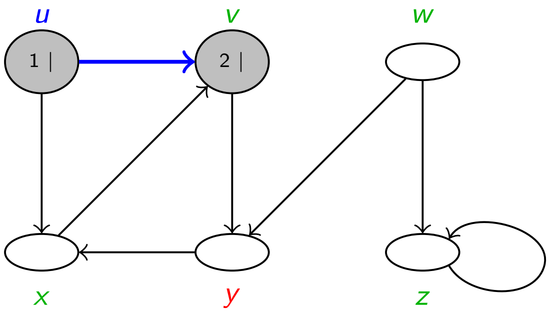
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

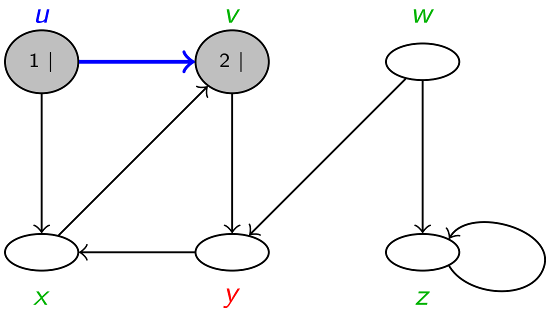
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



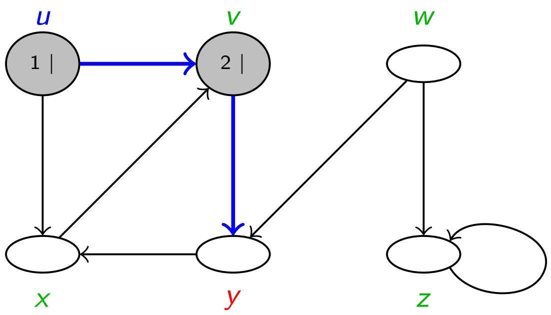
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

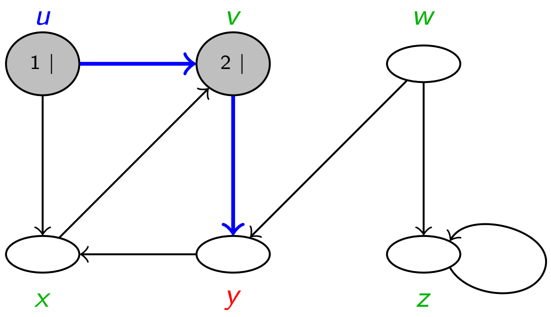
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;        // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$        // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



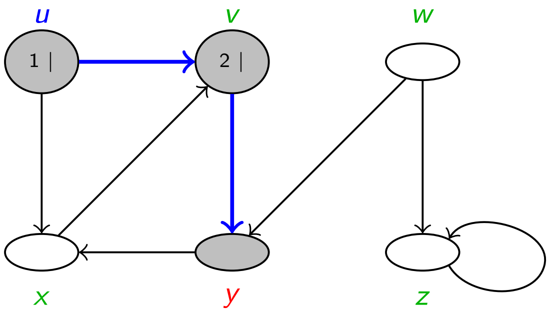
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



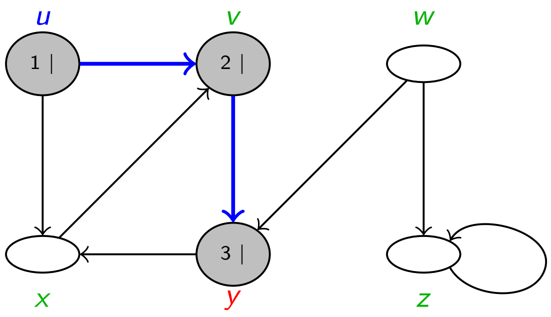
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] =$ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] =$ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

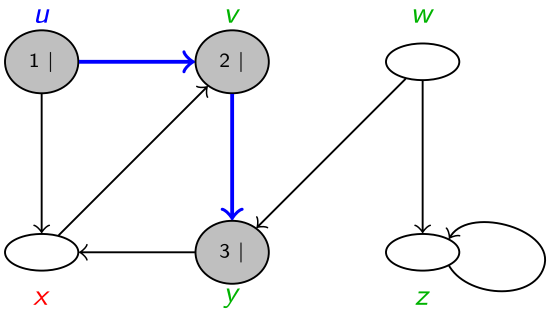
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;      // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$      // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

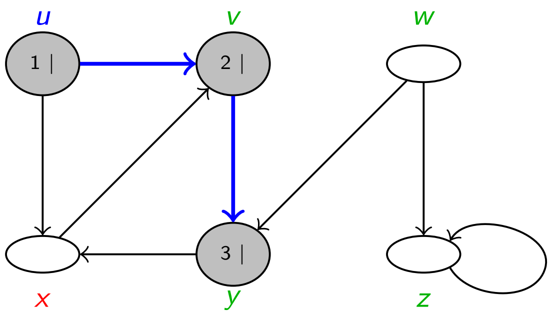
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



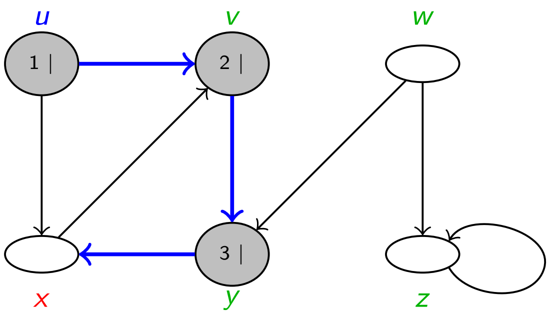
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] =$ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] =$ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



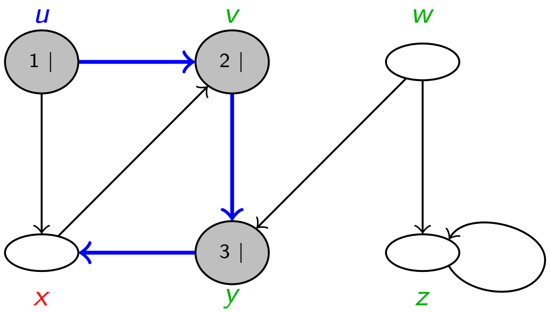
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;        // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$        // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

DFS(G)



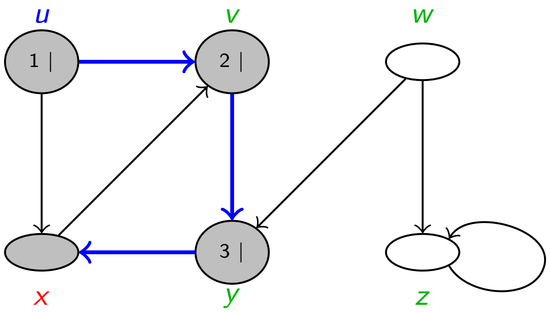
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

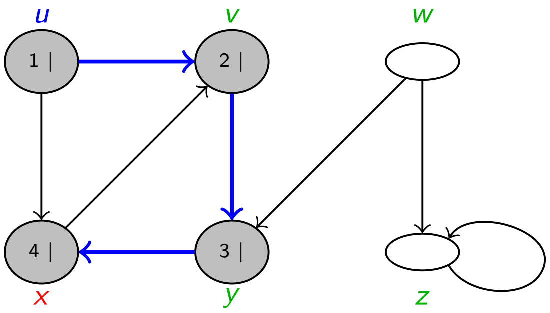
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

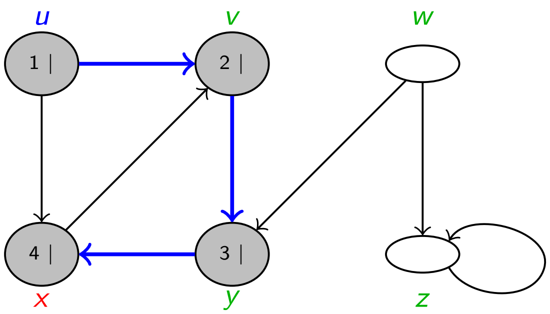
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

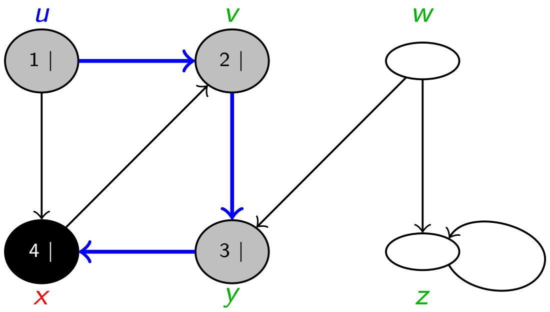
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
```

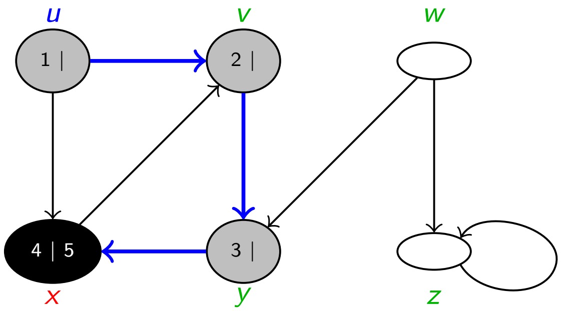
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
```

DFS(G)



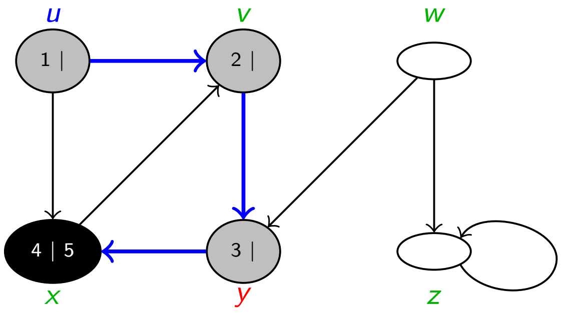
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



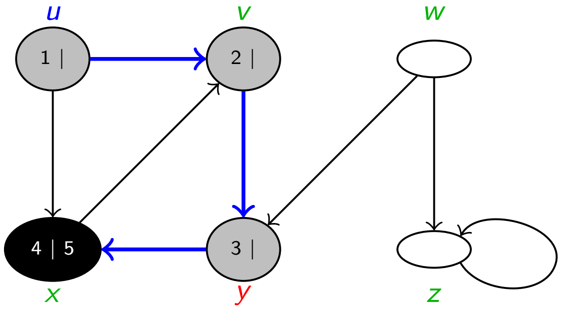
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



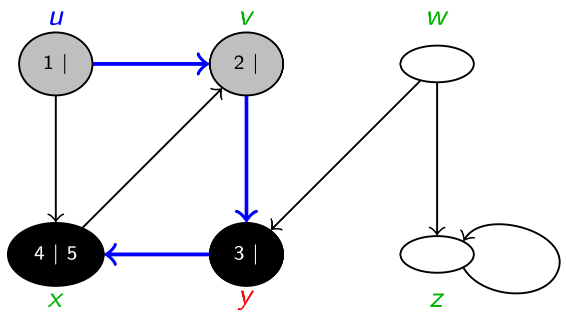
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



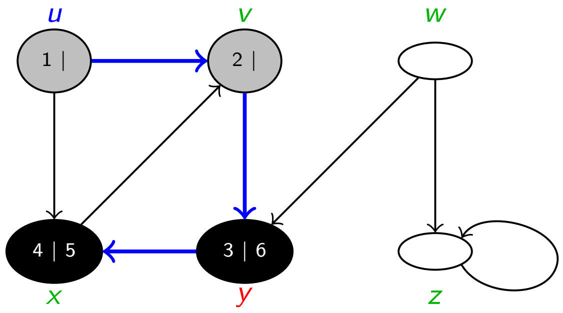
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = \mathrm{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \mathrm{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \mathrm{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \mathrm{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



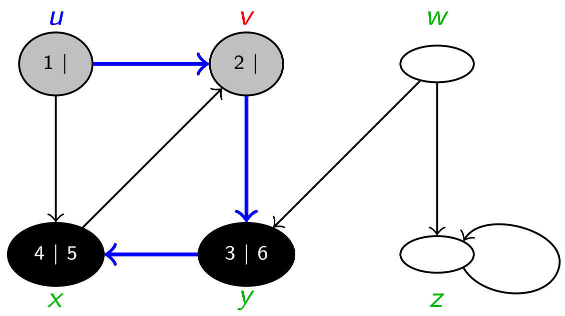
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if $(color[u] = \mathrm{WHITE})$
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \mathrm{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if $(color[v] = \mathrm{WHITE})$
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \mathrm{BLACK}$; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



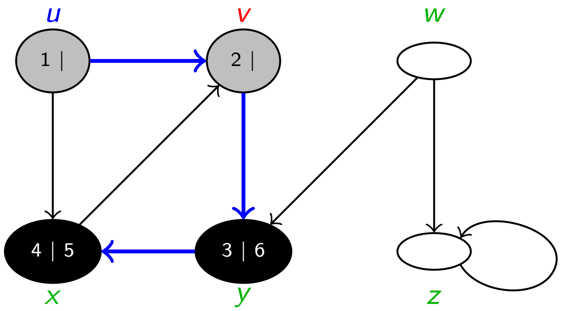
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if $(color[u] = \text{WHITE})$
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if $(color[v] = \text{WHITE})$
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



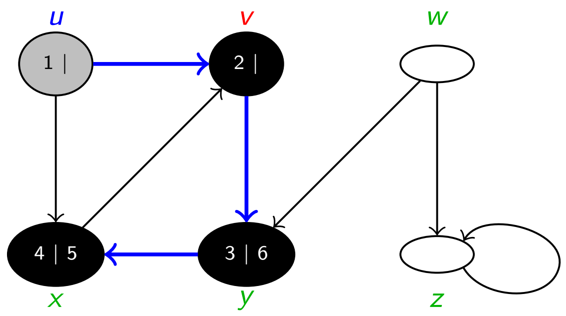
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

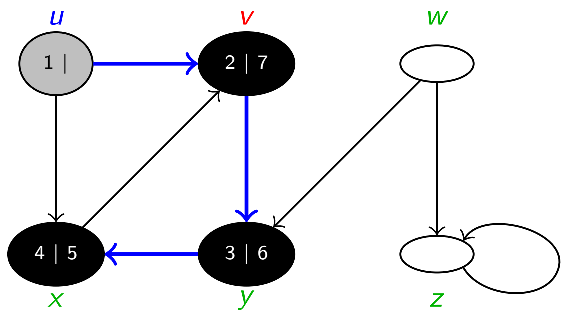
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



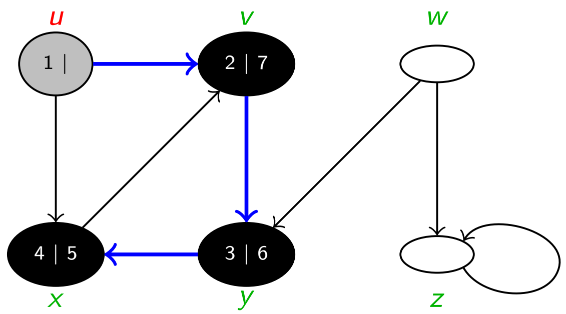
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

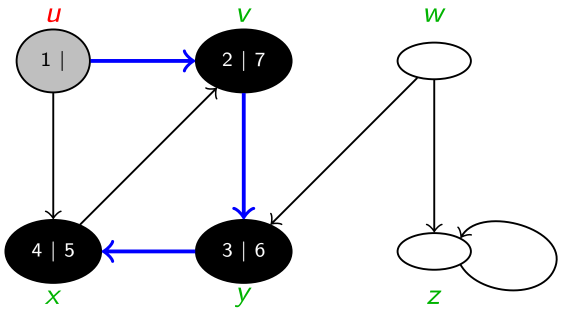
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



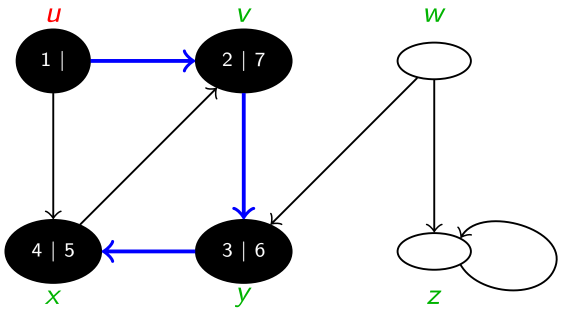
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

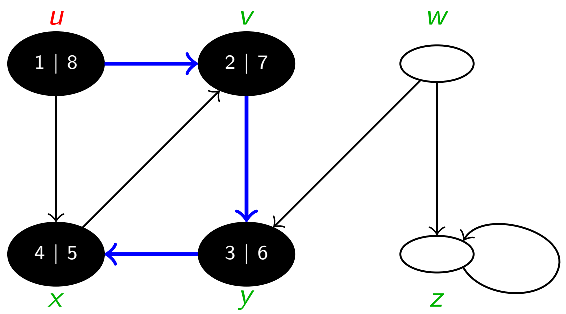
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

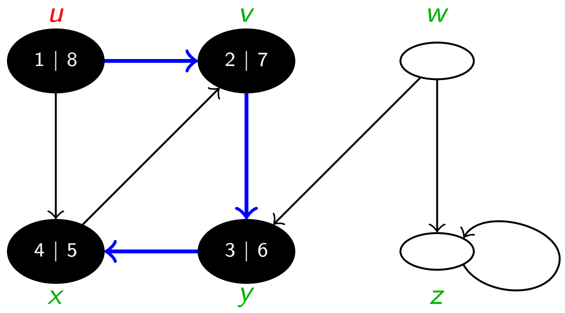
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \mathrm{WHITE}$)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \mathrm{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \mathrm{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \mathrm{BLACK}$; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



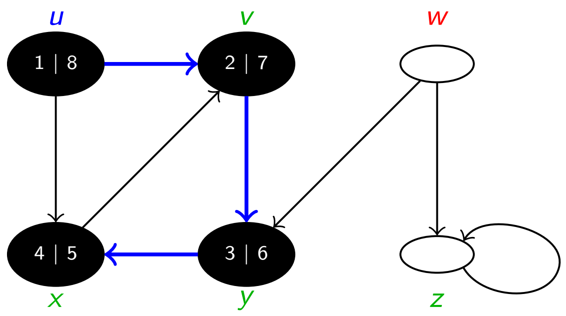
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \text{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



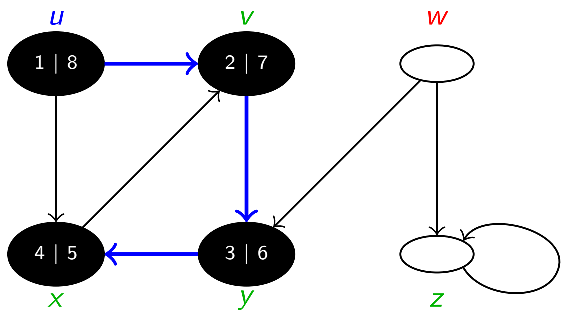
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] =$ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] =$ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



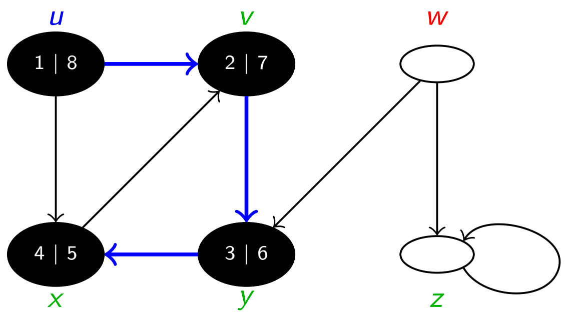
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



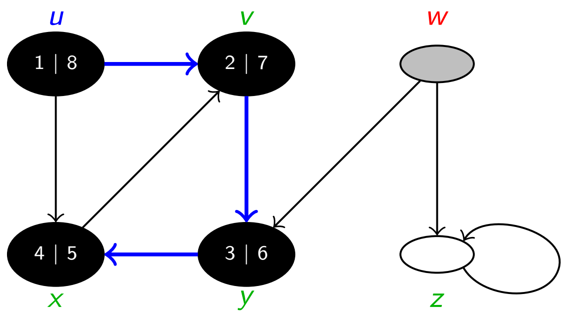
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \text{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



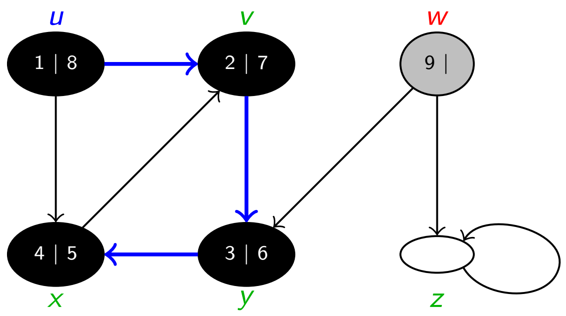
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \mathrm{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \mathrm{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \mathrm{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \mathrm{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



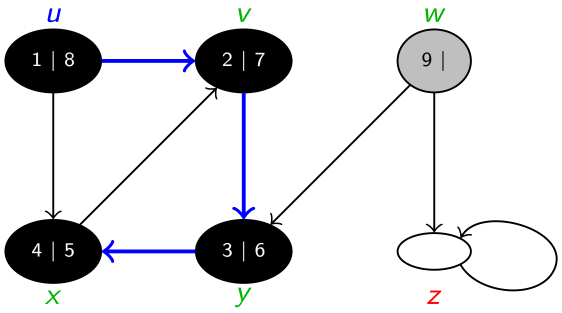
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



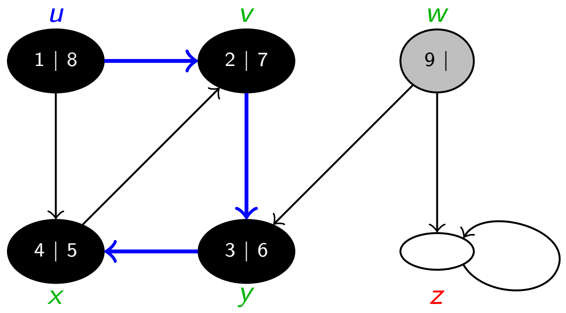
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \text{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

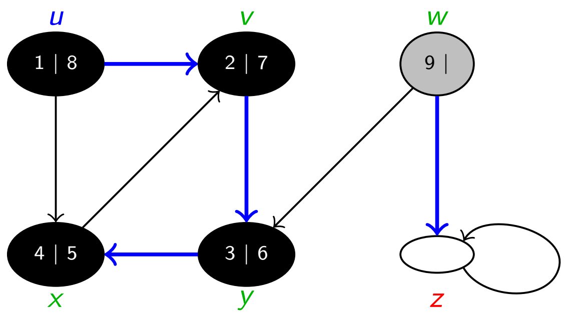
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End

DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



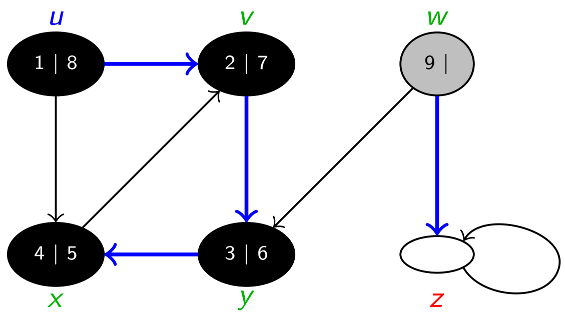
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



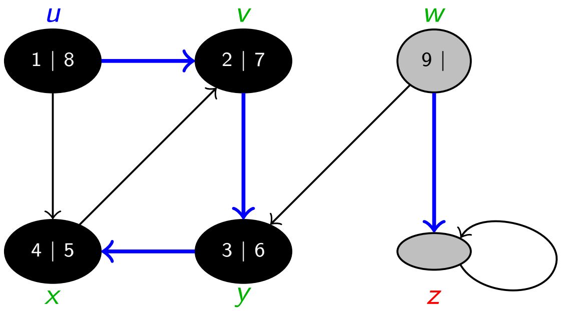
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \text{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \text{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \text{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \text{BLACK}$;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



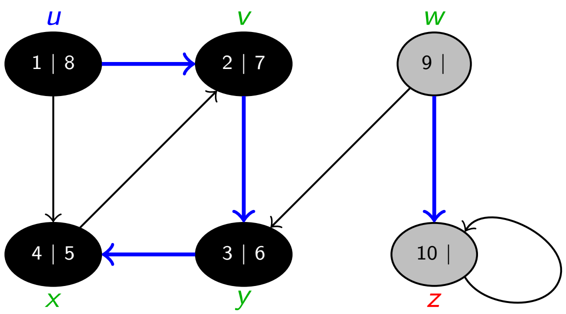
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
     . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



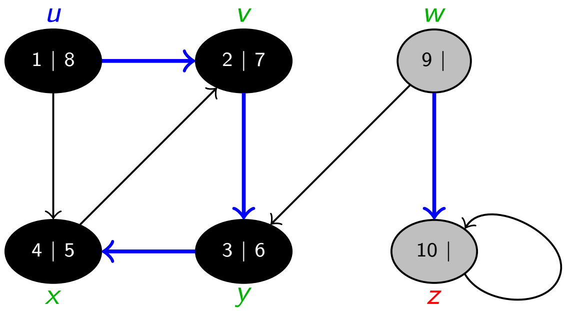
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



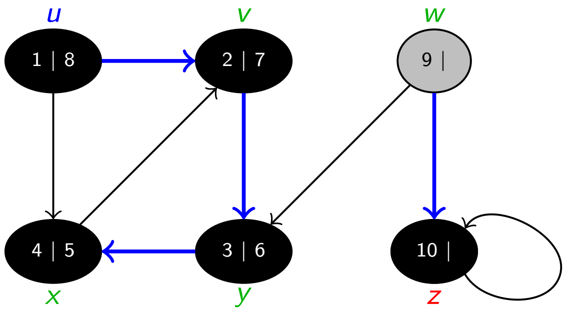
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



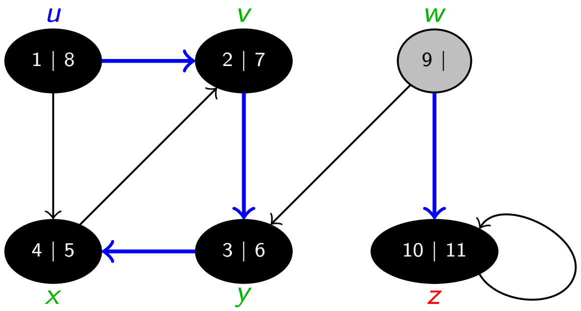
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



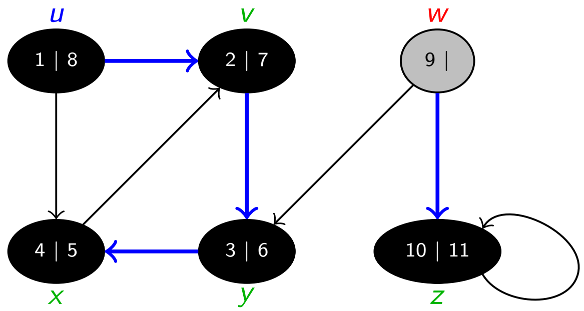
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



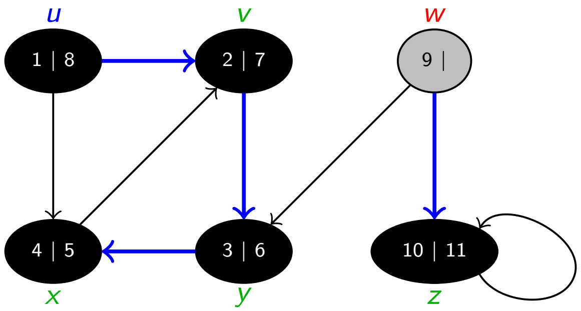
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if $(color[u] = \text{WHITE})$
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if $(color[v] = \text{WHITE})$
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

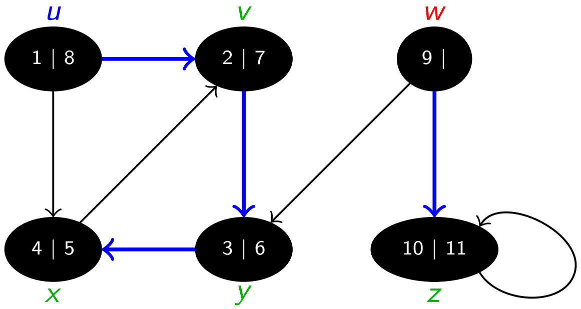
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



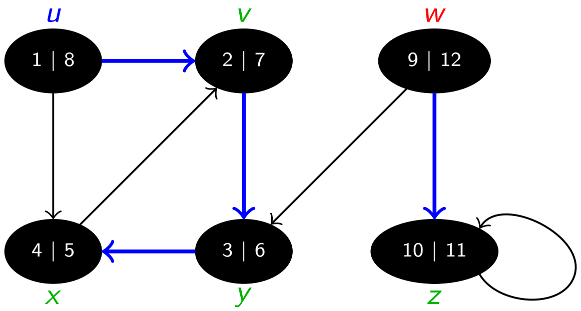
```
DFS($G$)
```

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

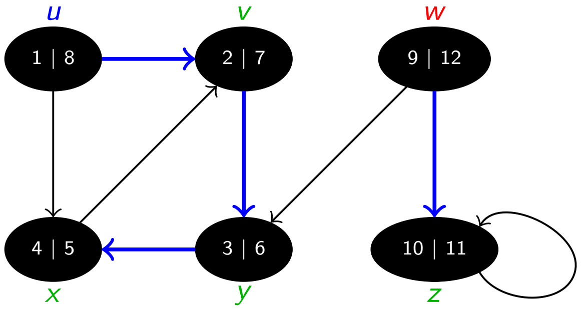
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = $ WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = $ WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



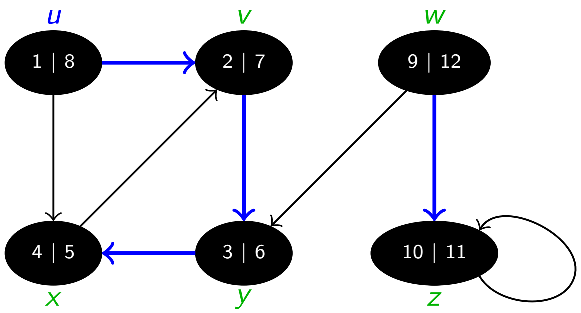
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    ...
    for each vertex $u \in V$
        if ($color[u] = \mathrm{WHITE}$)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow \mathrm{GRAY}$;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v] = \mathrm{WHITE}$)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow \mathrm{BLACK}$; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



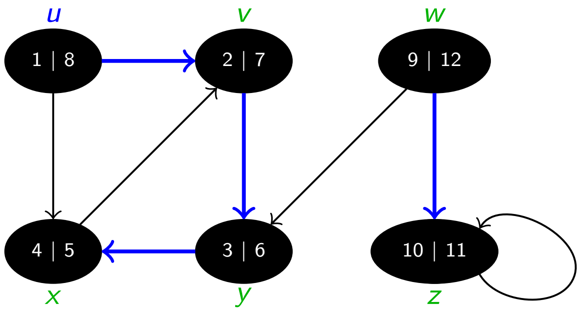
DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if $(color[u] = \text{WHITE})$
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if $(color[v] = \text{WHITE})$
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

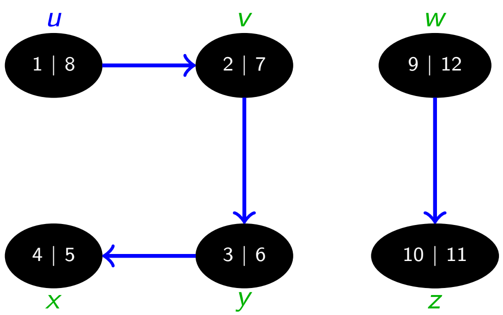
DFS(G)



```
DFS($G$)
I/P: $G = (V, E)$ in adjacency-list representation.
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;     // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$     // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK;  // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

DFS(G)



DFS Forest

DFS(G)

I/P: $G = (V, E)$ in adjacency-list representation.

```
Begin
    . . .
    for each vertex $u \in V$
        if ($color[u]$ = WHITE)
            DFS-VISIT($u$);
End
```

```
DFS-VISIT($u$)
Begin
    $color[u] \leftarrow$ GRAY;    // $u$ discovered.
    $time \leftarrow time + 1$;
    $d[u] \leftarrow time$;
    for each $v \in Adj[u]$    // Explore $(u, v)$.
        if ($color[v]$ = WHITE)
            $\pi[v] \leftarrow u$;
            DFS-VISIT($v$);
    $color[u] \leftarrow$ BLACK; // Blacken $u$ (finished).
    $f[m] \leftarrow time \leftarrow time + 1$;
End
```

Cost Analysis

- DFS:
    - **Initialization (Lines 1-3):** Takes $\Theta(|V|)$,
    - **Lines 5-7:** Takes $\Theta(|V|)$ (excluding $\mathrm{DFS\text{-}VISIT}$).

Cost Analysis

- DFS:
  - **Initialization (Lines 1-3):** Takes $\Theta(|V|)$,
  - **Lines 5-7:** Takes $\Theta(|V|)$ (excluding $\mathrm{DFS\text{-}VISIT}$).
- DFS-VISIT:
  - **Note:** $\mathrm{DFS\text{-}VISIT}$ is called exactly once for each $v \in V$.
    - $\mathrm{DFS\text{-}VISIT}$ is invoked only on WHITE vertices.
    - The first thing it does is paint the vertex GRAY.
  - $\mathrm{DFS\text{-}VISIT}(u)$: The loop on lines 4-7 runs $|Adj[u]|$ times.
  - $\therefore$ the total cost of executing lines 4-7 is

$$\sum_{v \in V} |Adj[v]| \;=\; \Theta(|E|).$$

Cost Analysis

- DFS:
  - **Initialization (Lines 1-3):** Takes $\Theta(|V|)$,
  - **Lines 5-7:** Takes $\Theta(|V|)$ (excluding $\mathrm{DFS\text{-}VISIT}$).
- DFS-VISIT:
  - **Note:** $\mathrm{DFS\text{-}VISIT}$ is called exactly once for each $v \in V$.
    - $\mathrm{DFS\text{-}VISIT}$ is invoked only on WHITE vertices.
    - The first thing it does is paint the vertex GRAY.
  - $\mathrm{DFS\text{-}VISIT}(u)$: The loop on lines 4-7 runs $|Adj[u]|$ times.
  - $\therefore$ the total cost of executing lines 4-7 is

$$\sum_{v \in V} |Adj[v]| \ = \ \Theta(|E|).$$

- **Total Complexity:** $\Theta(|V| + |E|)$.

Classification of Edges

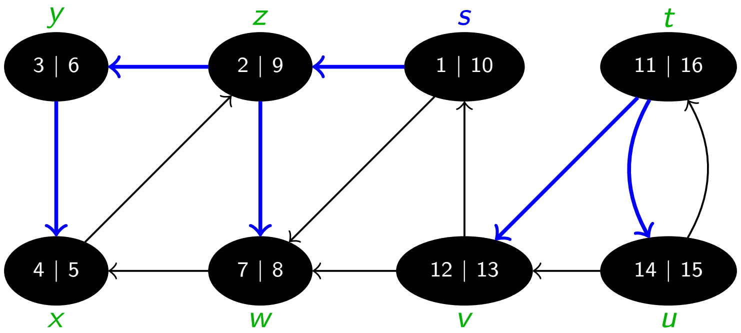
Edge Types: Directed Graphs

Four types of edges are defined w.r.t. a depth-first forest G_π:

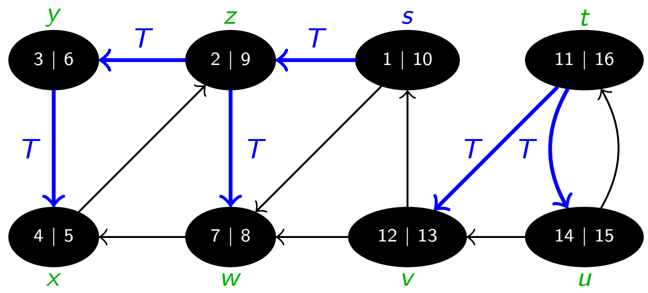
1. Tree edges: These are edges in the depth-first forest G_π.
2. Back edges: Edges (u, v) connecting u to it's ancestor v.

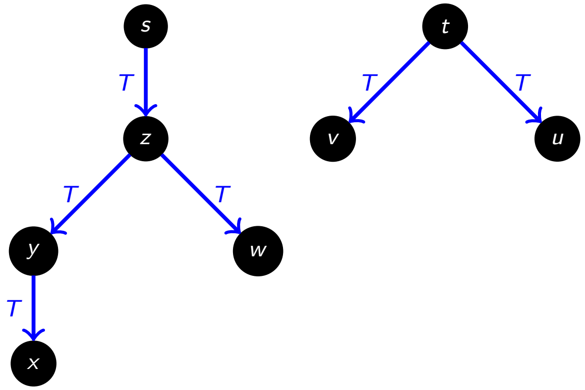
   Self-loops in directed graphs are considered as back edges.
3. Forward edges: Are those nontree edges (u, v) that connect a vertex u to a descendant v.
4. Cross edges: Are all other edges.
   - They can go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or
   - they can go between vertices in different depth-first trees.

An Example



An Example

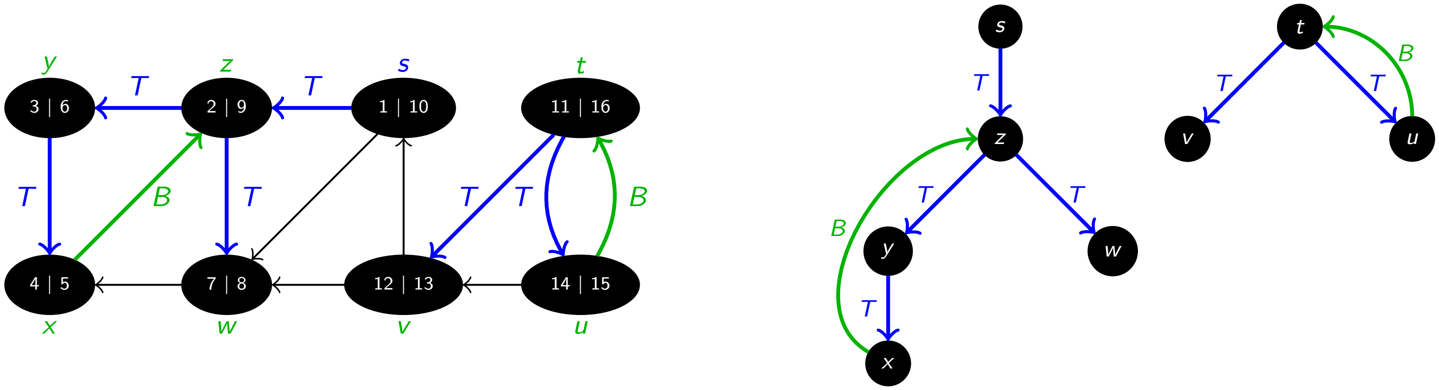




Key idea: Each edge (u, v) can be classified by the color of the vertex v that is reached when the edge is first explored (except that forward and cross edges are not distinguished).

- Tree edges (T): If the colour of v is WHITE.

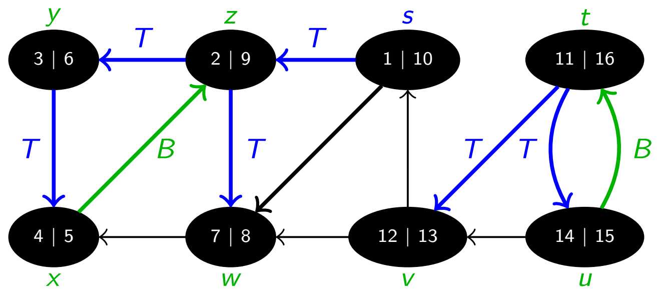
An Example

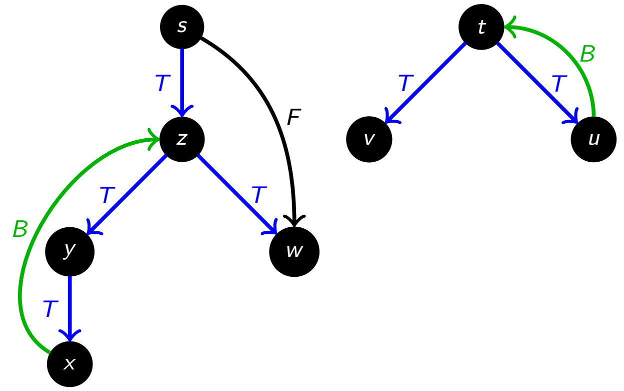


Key idea: Each edge (u, v) can be classified by the color of the vertex v that is reached when the edge is first explored (except that forward and cross edges are not distinguished).

- Tree edges (T): If the colour of v is WHITE.
- Back edges (B): If the colour of v is GRAY.

An Example

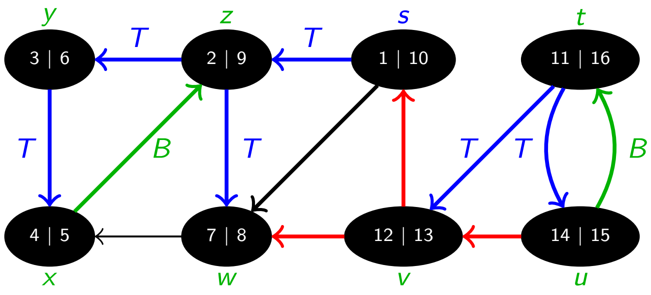


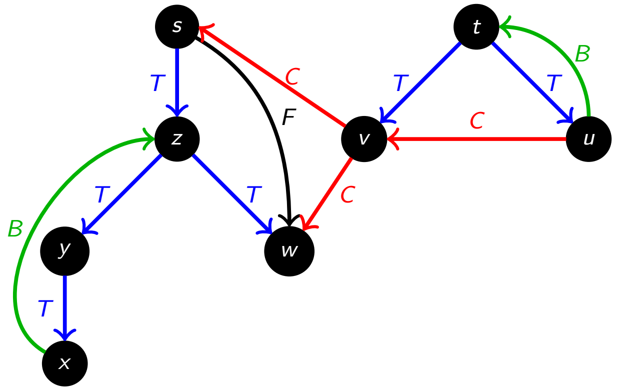


Key idea: Each edge (u, v) can be classified by the color of the vertex v that is reached when the edge is first explored (except that forward and cross edges are not distinguished).

- Tree edges (T): If the colour of v is WHITE.
- Back edges (B): If the colour of v is GRAY.
- Forward edges (F): If the colour of v is BLACK.

An Example





Key idea: Each edge (u, v) can be classified by the color of the vertex v that is reached when the edge is first explored (except that forward and cross edges are not distinguished).

- Tree edges (T): If the colour of v is WHITE.
- Back edges (B): If the colour of v is GRAY.
- Forward edges (F): If the colour of v is BLACK.
- Cross edges (C): If the colour of v is BLACK.

Books and Other Materials Consulted

1. Definitions taken from Discrete Mathematics Lecture Notes (M. Tech (CS), Monsoon Semester, 2007) taught by Prof. Palash Sarkar (ASU, ISI Kolkata).

2. DFS taken from *Introduction to Algorithms* by Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.

Thank You for your kind attention!

Questions!!