DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

DEPTH-FIRST SEARCH

AGENDA

- Depth-first search algorithm
  - Timestamp
  - Depth-first trees (forest)
  - Parenthesized expression

DEPTH-FIRST SEARCH (DFS)

- The strategy is to ***search "deeper" in the graph*** whenever possible
- It ***explores edges*** out of the most recently discovered vertex v that still has unexplored edges leaving it
- If any undiscovered vertices remain, then depth-first search selects one of them as a new source.
- The algorithm repeats this entire process until it has discovered every vertex.

THE DFS ALGORITHM

- Input
  - a graph $G = (V, E)$ that is represented BY **adjacency lists**
- The algorithm selects an undiscovered vertex as a new source.
- Then it uses a **DFS-VISIT** procedure to ***explores edges***.

DFS (G)
for each vertex $u \in G.V$
$u.\boldsymbol{color}$ = WHITE
$u.\boldsymbol{\pi}$ = NIL
$time = 0$
for each vertex $u \in G.V$
if $u.color$ == WHITE
DFS-VISIT (G, u)

THE DFS ALGORITHM
THE VERTEX OBJECT

- For each vertex $\boldsymbol{u} \in V$,
  - $\boldsymbol{u.d}$ – the time vertex u is ***discovered***.
  - $\boldsymbol{u.f}$ – the time vertex u is ***finished***.
  - $\boldsymbol{u.color}$ – distinguish between *discovered* and *undiscovered* vertices. Vertex u is
    - **White** before time $\boldsymbol{u.d}$,
    - **Gray** between time $\boldsymbol{u.d}$ and $\boldsymbol{u.f}$, and
    - **Black** thereafter.
  - $\boldsymbol{u.\pi}$ – the ***predecessor*** of vertex u.
    - If u has no predecessor, then $u.\pi = \mathtt{NIL}$.

```
DFS (G)
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G,u)
```

```
DFS-VISIT (G,u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                u.π = u
                DFS-VISIT (G,v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM TIMESTAMP

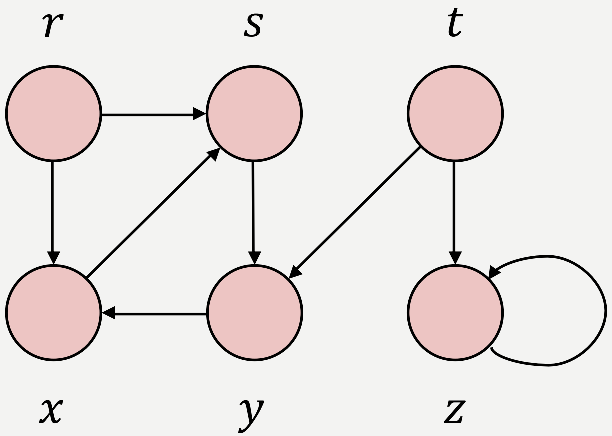
- The $time$ variable is a global timestamp.
- The algorithm ***timestamps*** each vertex:
  - The $\boldsymbol{u.d}$ records when u is first ***discovered*** (and **grayed**), and
  - the $\boldsymbol{u.f}$ records when the search ***finishes*** examining u's adjacency list (and blackens u).
  - For each node u, $\boldsymbol{u.d} < \boldsymbol{u.f}$.
- These timestamps are integers **between 1 and 2$|V|$,** since there is **one** ***discovery*** event and **one** ***finishing*** event for each of the $|V|$ vertices.

DFS (G)
for each vertex $u \in G.V$
$\quad u.color =$ WHITE
$\quad u.\pi =$ NIL
$\boldsymbol{time} = 0$
for each vertex $u \in G.V$
$\quad$ **if** $u.color ==$ WHITE
$\quad\quad$ DFS-VISIT (G, u)

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$
$\boldsymbol{time} = \boldsymbol{time} + 1$
$\boldsymbol{u.d} = \boldsymbol{time}$
$u.color =$ GRAY
for each $v \in G.Adj[u]$
$\quad$ **if** $v.color ==$ WHITE
$\quad\quad v.\pi = u$
$\quad\quad$ DFS-VISIT (G, v)
$u.color =$ BLACK
$\boldsymbol{time} = \boldsymbol{time} + 1$
$\boldsymbol{u.f} = \boldsymbol{time}$

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

- Global timer: $time =$

Stack

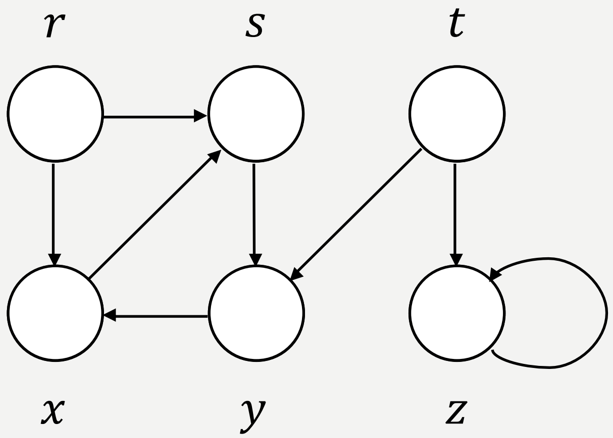
Stack is used to show the working process of DFS-VISIT algorithm

DFS $(\boldsymbol{G})$
for each vertex $u \in G.V$
$u.color =$ WHITE
$u.\pi =$ NIL
$time = 0$
for each vertex $u \in G.V$
if $u.color ==$ WHITE
DFS-VISIT (G, u)

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$
$time = time + 1$
$u.d = time$
$u.color =$ GRAY
for each $v \in G.Adj[u]$
if $v.color ==$ WHITE
$v.\pi = u$
DFS-VISIT (G, v)
$u.color =$ BLACK
$time = time + 1$
$u.f = time$

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

Stack

- Global timer: $time = \mathbf{0}$

DFS (G)

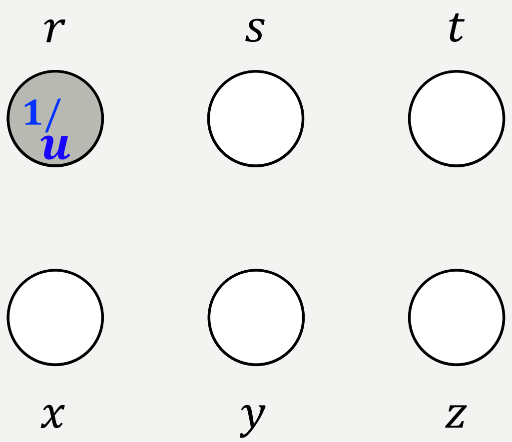
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$\boxed{r} \rightarrow s \rightarrow x$

$s \rightarrow y$

$t \rightarrow y \rightarrow z$

$x \rightarrow s$

$y \rightarrow x$

$z \rightarrow z$

- Global timer: $time = \mathbf{1}$

DFS-VISIT$(G, \boldsymbol{r})$

Stack

DFS (G)

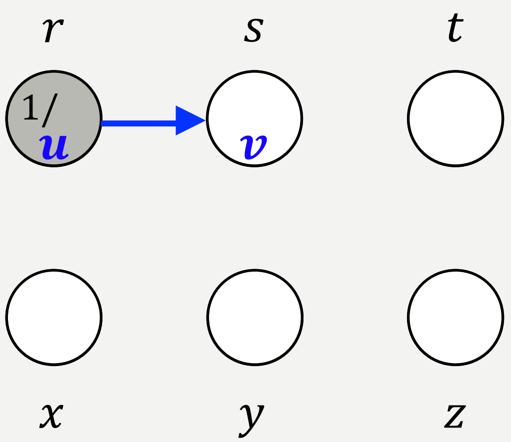
for each vertex $u \in G.V$
    $u.color =$ WHITE
    $u.\pi =$ NIL
$time = 0$
for each vertex $u \in G.V$
    if $u.color ==$ WHITE
        DFS-VISIT (G, u)

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$

$time = time + 1$
$u.d = time$
$u.color =$ GRAY
for each $v \in G.Adj[u]$
    if $v.color ==$ WHITE
        $v.\pi = u$
        DFS-VISIT (G, v)
$u.color =$ BLACK
$time = time + 1$
$u.f = time$

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$\boxed{r} \to s \to x$

$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

- Global timer: $time = \mathbf{1}$

Stack
DFS-VISIT(G, s)
DFS-VISIT(G, r)

DFS (G)

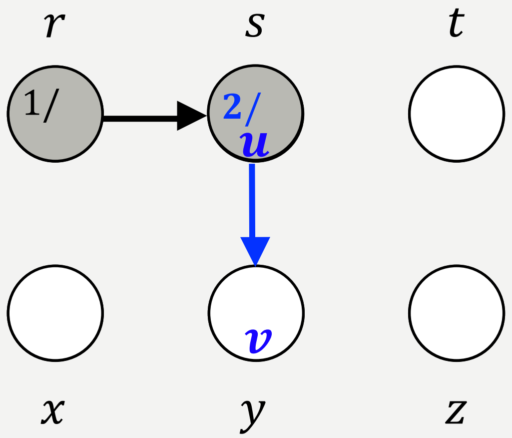
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to \boxed{s} \to x$

$\boxed{s} \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

DFS-VISIT(G, y)
DFS-VISIT(G, s)
DFS-VISIT(G, r)

Stack

- Global timer: $time = \mathbf{2}$

DFS (G)

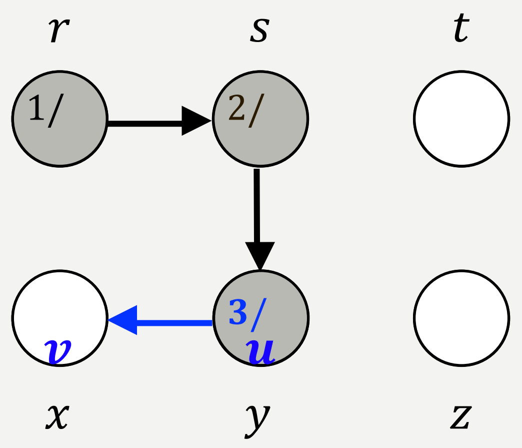
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to \boxed{s}^{1} \to x$

$s \to \boxed{y}^{2}$

$t \to y \to z$

$x \to s$

$\boxed{y}^{3} \to x$

$z \to z$

DFS-VISIT$(G, \boldsymbol{x})$

DFS-VISIT(G, y)

DFS-VISIT(G, s)

DFS-VISIT(G, r)

Stack

- Global timer: $time = \mathbf{3}$

DFS $(\boldsymbol{G})$

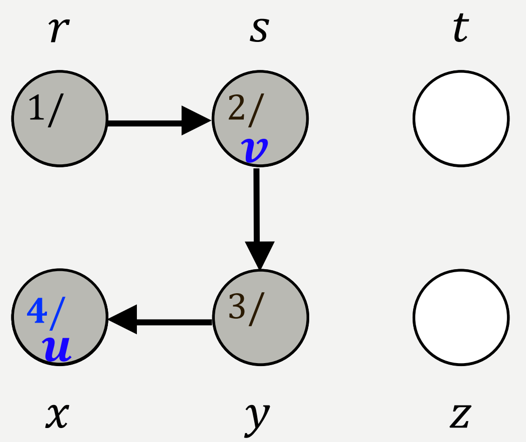
```
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G, u)
```

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                v.π = u
                DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to \boxed{s}^{1} \to x$

$s \to \boxed{y}^{2}$

$t \to y \to z$

$\boxed{x}^{4} \to s$

$y \to \boxed{x}^{3}$

$z \to z$

Stack
DFS-VISIT(G, x)
DFS-VISIT(G, y)
DFS-VISIT(G, s)
DFS-VISIT(G, r)

Stack

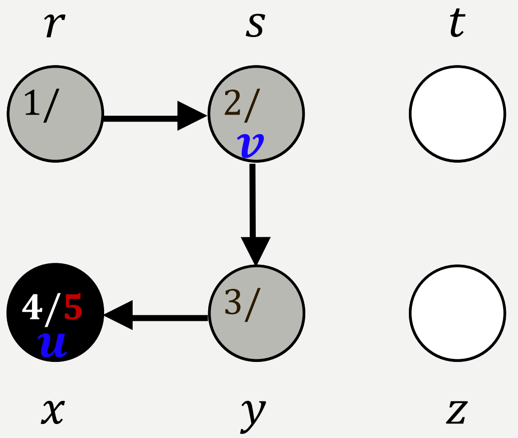
- Global timer: $time = \mathbf{4}$

```
DFS (G)
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

```
DFS-VISIT (G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \rightarrow \boxed{s}^{1} \rightarrow x$

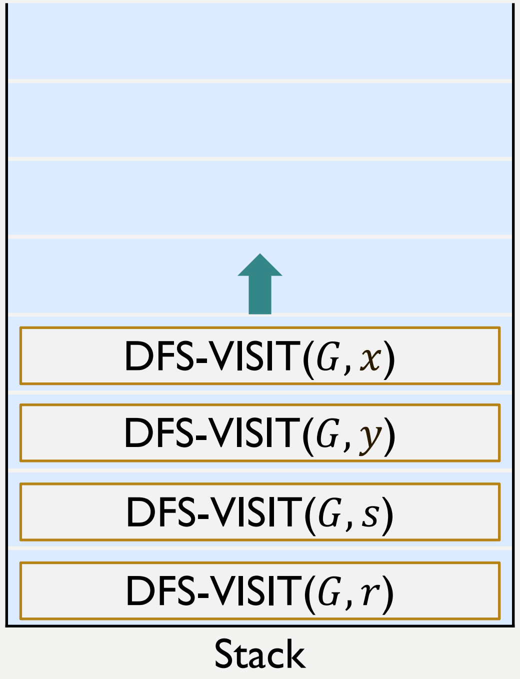
$s \rightarrow \boxed{y}^{2}$

$t \rightarrow y \rightarrow z$

$x \rightarrow s$ $\boxed{}^{4}$

$y \rightarrow \boxed{x}^{3}$

$z \rightarrow z$



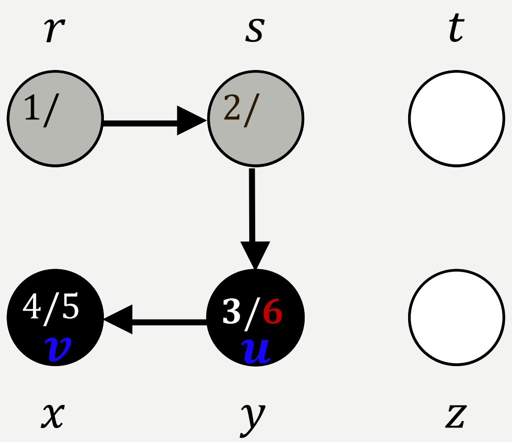
- Global timer: $time = \mathbf{5}$

DFS (G)
for each vertex $u \in G.V$
$\quad u.color$ = WHITE
$\quad u.\pi$ = NIL
$time = 0$
for each vertex $u \in G.V$
$\quad$ **if** $u.color$ == WHITE
$\quad\quad$ DFS-VISIT (G, u)

DFS-VISIT (G, u)
$time = time + 1$
$u.d = time$
$u.color$ = GRAY
for each $v \in G.Adj[u]$
$\quad$ **if** $v.color$ == WHITE
$\quad\quad v.\pi = u$
$\quad\quad$ DFS-VISIT (G, v)
$u.color$ = BLACK
$time = time + 1$
$u.f = time$

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \rightarrow \boxed{s}^{1} \rightarrow x$

$s \rightarrow \boxed{y}^{2}$

$t \rightarrow y \rightarrow z$

$x \rightarrow s$

$y \rightarrow \boxed{x}^{3}$

$z \rightarrow z$

Stack
DFS-VISIT(G, y)
DFS-VISIT(G, s)
DFS-VISIT(G, r)

- Global timer: $time = \mathbf{6}$

DFS (G)

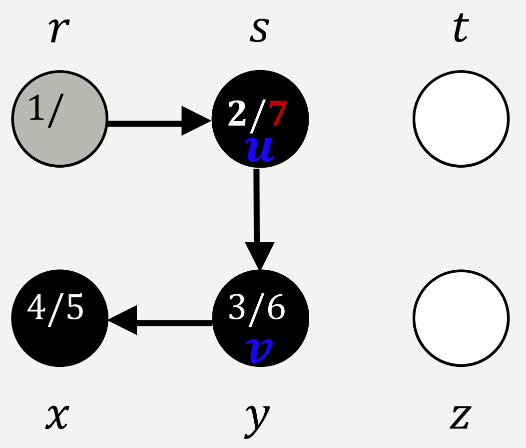
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to \boxed{s}^{1} \to x$

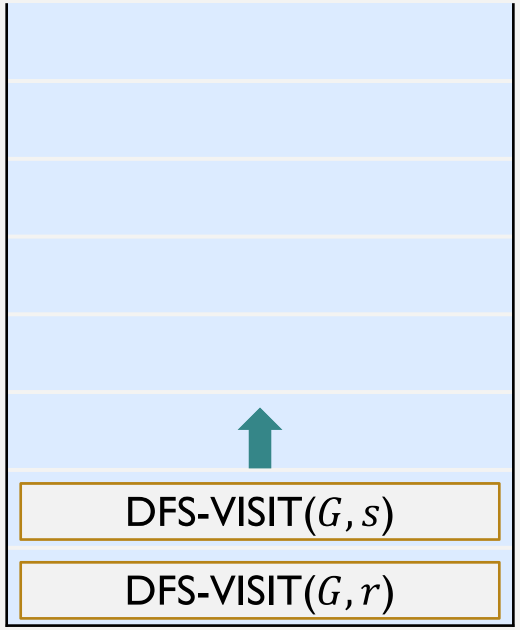
$s \to \boxed{y}^{2}$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$



Stack

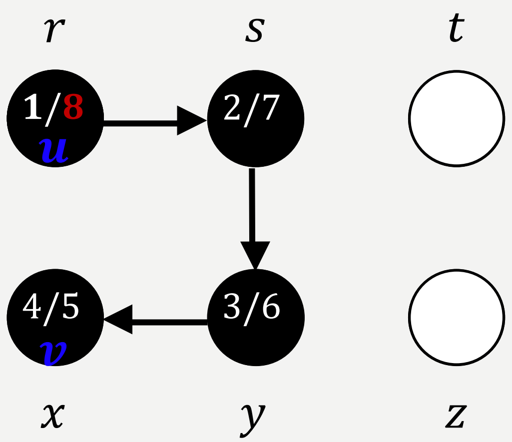
- Global timer: $time = \mathbf{7}$

DFS (G)
for each vertex $u \in G.V$
$\quad u.color =$ WHITE
$\quad u.\pi =$ NIL
$time = 0$
for each vertex $u \in G.V$
$\quad$ **if** $u.color ==$ WHITE
$\quad\quad$ DFS-VISIT (G, u)

DFS-VISIT (G, u)
$time = time + 1$
$u.d = time$
$u.color =$ GRAY
for each $v \in G.Adj[u]$
$\quad$ **if** $v.color ==$ WHITE
$\quad\quad v.\pi = u$
$\quad\quad$ DFS-VISIT (G, v)
$u.color =$ BLACK
$time = time + 1$
$u.f = time$

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to \boxed{s} \to x$

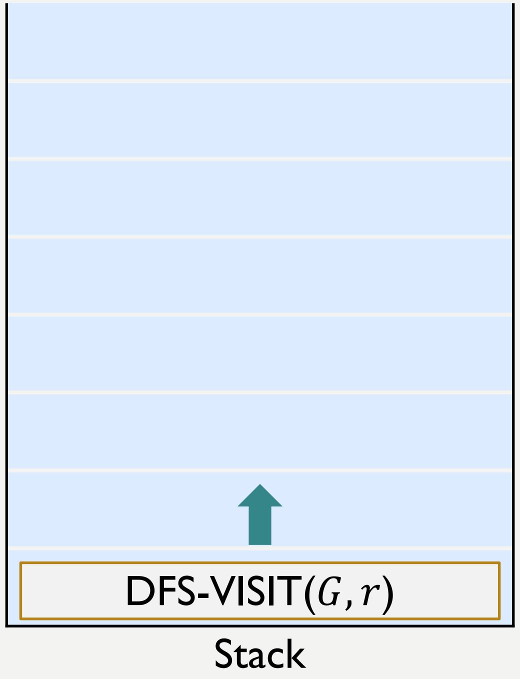
$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$



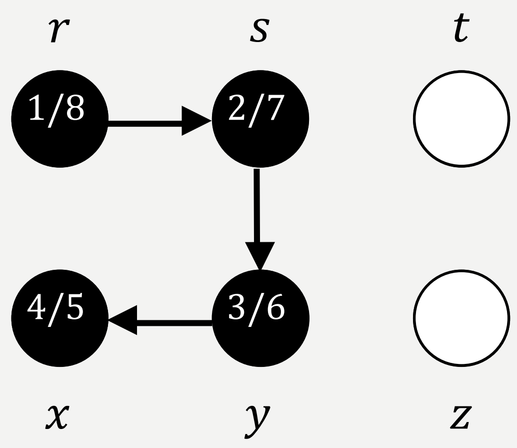
- Global timer: $time = \mathbf{8}$

```
DFS (G)
for each vertex u ∈ G.V
      u.color = WHITE
      u.π = NIL
time = 0
for each vertex u ∈ G.V
      if u.color == WHITE
            DFS-VISIT (G, u)
```

```
DFS-VISIT (G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
      if v.color == WHITE
            v.π = u
            DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

- Global timer: $time = \mathbf{8}$

DFS-VISIT$(G, \boldsymbol{t})$

Stack

DFS $(\boldsymbol{G})$

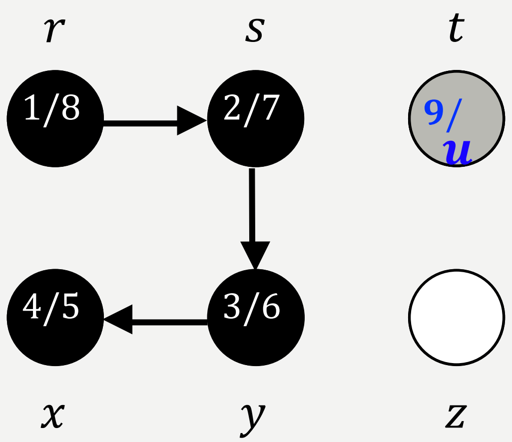
for each vertex $u \in G.V$
$\quad u.color$ = WHITE
$\quad u.\pi$ = NIL
$time = 0$
for each vertex $u \in G.V$
$\quad$ **if** $u.color ==$ WHITE
$\qquad$ DFS-VISIT (G, u)

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$

$time = time + 1$
$u.d = time$
$u.color$ = GRAY
for each $v \in G.Adj[u]$
$\quad$ **if** $v.color ==$ WHITE
$\qquad v.\pi = u$
$\qquad$ DFS-VISIT (G, v)
$u.color$ = BLACK
$time = time + 1$
$u.f = time$

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$\boxed{t} \to y \to z$

$x \to s$

$y \to x$

$z \to z$

- Global timer: $time = \mathbf{9}$

Stack
DFS-VISIT(G, t)

DFS (G)

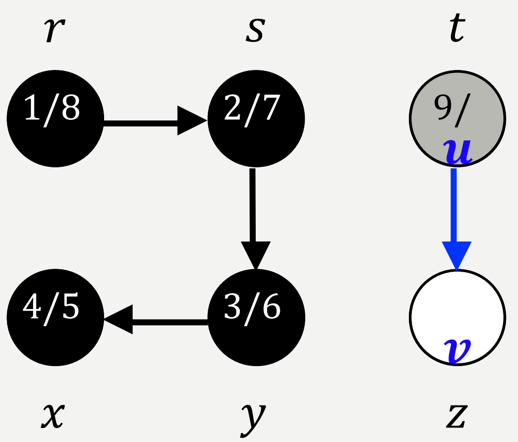
```
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                v.π = u
                DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \rightarrow s \rightarrow x$

$s \rightarrow y$

$t \rightarrow y \rightarrow z$

$x \rightarrow s$

$y \rightarrow x$

$z \rightarrow z$

- Global timer: $time = \mathbf{9}$

DFS-VISIT(G, z)

DFS-VISIT(G, t)

Stack

DFS (G)

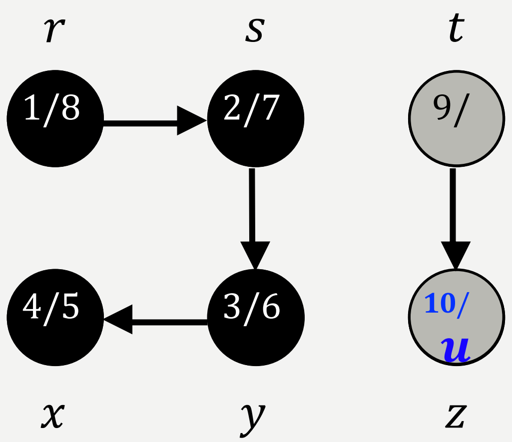
```
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                v.π = u
                DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$t \to y \to \boxed{z}$

$x \to s$

$y \to x$

$\boxed{z} \to z$

- Global timer: $time = \mathbf{10}$

DFS-VISIT(G, z)

DFS-VISIT(G, t)

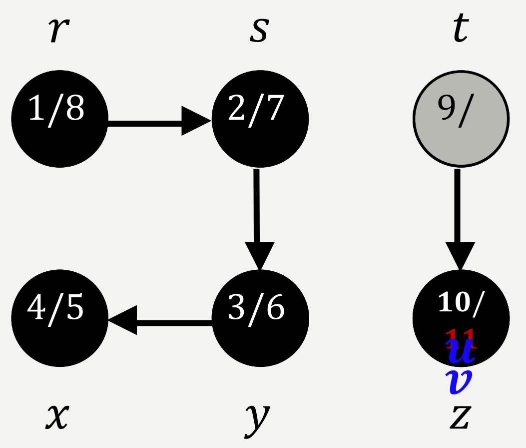
Stack

```
DFS (G)
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G, u)
```

```
DFS-VISIT (G, u)
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                v.π = u
                DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

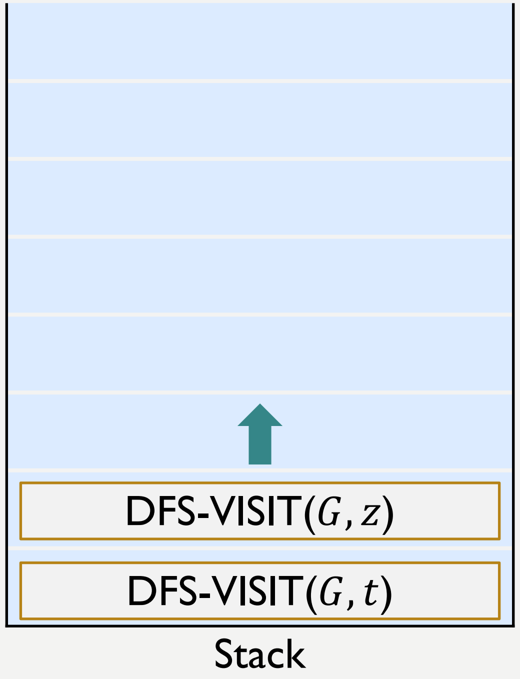
$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$



- Global timer: $time = \mathbf{11}$

DFS (G)

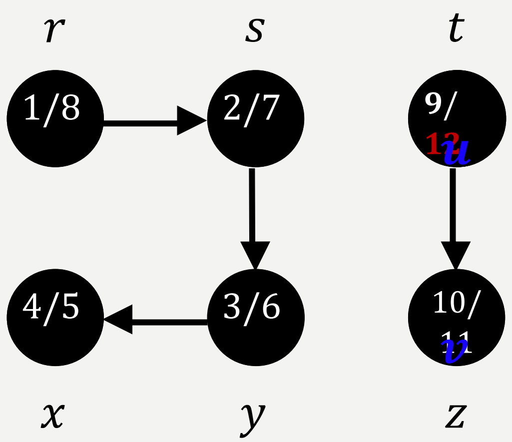
```
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
        if v.color == WHITE
                v.π = u
                DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

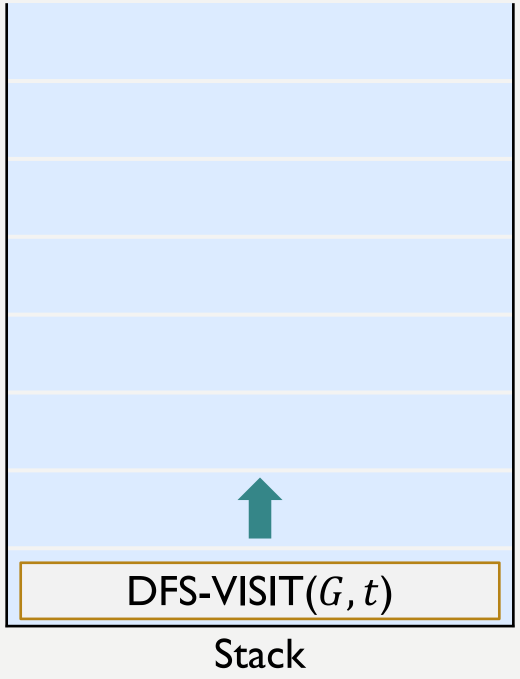
$t \to y \to \boxed{z}$

$x \to s$

$y \to x$

$z \to z$

- Global timer: $time = \mathbf{12}$



DFS (G)

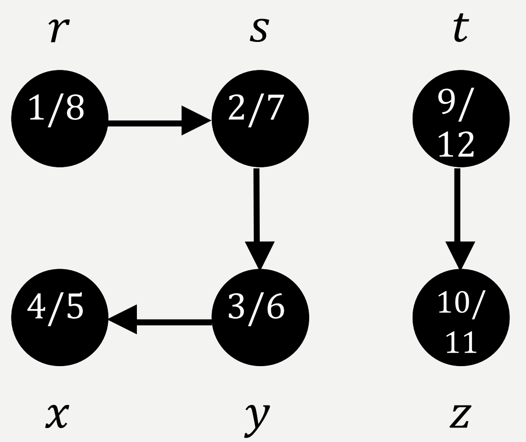
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM IN ACTION

- Apply the algorithm on the graph below.



Adjacency list

$r \to s \to x$

$s \to y$

$t \to y \to z$

$x \to s$

$y \to x$

$z \to z$

Stack

- Global timer: $time = \mathbf{12}$

DFS (G)

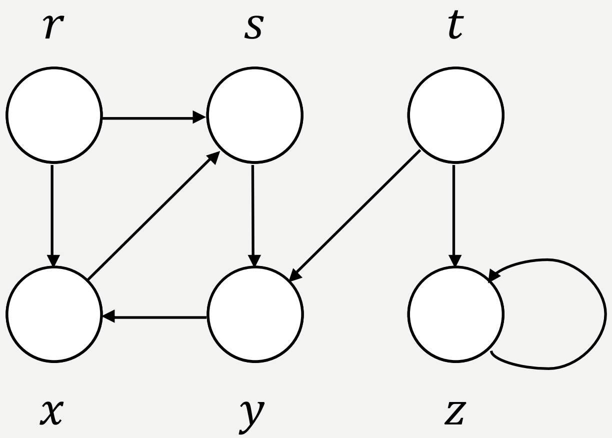
```
for each vertex u ∈ G.V
    u.color = WHITE
    u.π = NIL
time = 0
for each vertex u ∈ G.V
    if u.color == WHITE
        DFS-VISIT (G, u)
```

DFS-VISIT (G, u)

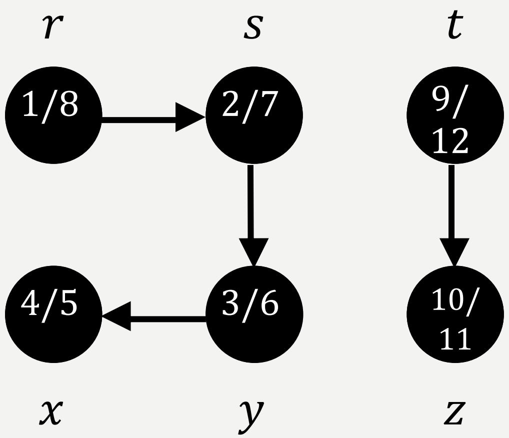
```
time = time + 1
u.d = time
u.color = GRAY
for each v ∈ G.Adj[u]
    if v.color == WHITE
        v.π = u
        DFS-VISIT (G, v)
u.color = BLACK
time = time + 1
u.f = time
```

THE DFS ALGORITHM
THE RESULT

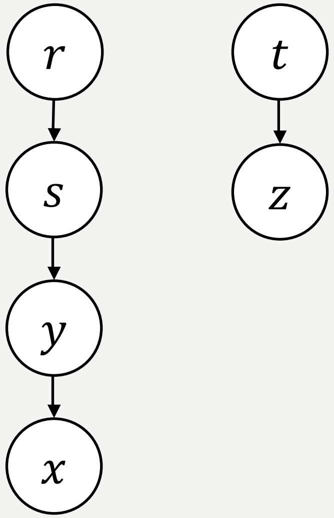
- Let $G_\pi = (V, E_\pi)$, where $E_\pi = \{(v.\pi, v): v \in V \text{ and } v.\pi \neq NIL\}$, then G_π is the resulting ***predecessor* subgraph** from running the DFS.
- The predecessor subgraph of a DFS forms a ***depth-first forest*** comprising several ***depth-first trees***. The edges in E_π are ***tree edges***.



Original



Predecessor subgraph (depth-first forest)



Tree visualization

THE DFS ALGORITHM RUNNING TIME - INIT.

- Initialization of DFS (G)
  - Line 1 ~ 3 executes $\Theta($ __________ $)$ time(s).
  - Line 5 ~ 7 executes $\Theta($ __________ $)$ time(s).
  - The running time of DFS(G), ***exclusive of*** the time to execute the calls to DFS-VISIT is $\Theta($ __________ $)$.

	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)

THE DFS ALGORITHM
RUNNING TIME – DFS-VISIT

- Execution of DFS-VISIT (G, u)
  - For each vertex $v \in V$, DFS-VISIT(G, u) is called ______ time(s).
  - During an execution of DFS-VISIT(G, u), the loop on lines 4-7 executes $\Theta($ ___________ $)$ times.
  - The property of adjacency list

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

the total cost of executing lines 4-7 of DFS-VISIT is $\Theta($ ________ $)$.

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

THE DFS ALGORITHM
RUNNING TIME – OVERALL

- Conclude
  - The running time of DFS(G), ***exclusive of*** the time to execute the calls to DFS-VISIT is $\boldsymbol{\Theta}(|\boldsymbol{V}|)$.
  - The total cost of executing lines 4-7 of DFS-VISIT is $\boldsymbol{\Theta}(|\boldsymbol{E}|)$.

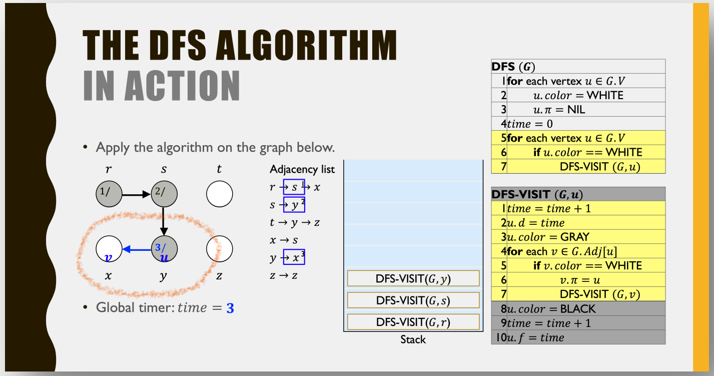
- The running time of the entire DFS procedure is $\Theta($ __________ $)$.

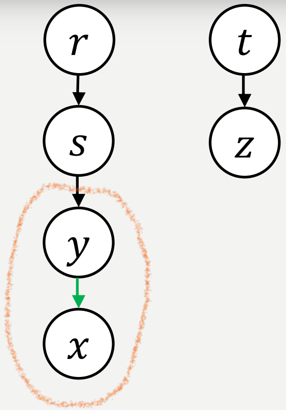
DFS $(\boldsymbol{G})$	
1	**for** each vertex $u \in G.V$
2	$\quad u.color =$ WHITE
3	$\quad u.\pi =$ NIL
4	$time = 0$
5	**for** each vertex $u \in G.V$
6	$\quad$ **if** $u.color ==$ WHITE
7	$\quad\quad$ DFS-VISIT (G, u)

DFS-VISIT $(\boldsymbol{G}, \boldsymbol{u})$	
1	$time = time + 1$
2	$u.d = time$
3	$u.color =$ GRAY
4	**for** each $v \in G.Adj[u]$
5	$\quad$ **if** $v.color ==$ WHITE
6	$\quad\quad v.\pi = u$
7	$\quad\quad$ DFS-VISIT (G, v)
8	$u.color =$ BLACK
9	$time = time + 1$
10	$u.f = time$

THE PROPERTY OF DFS #1

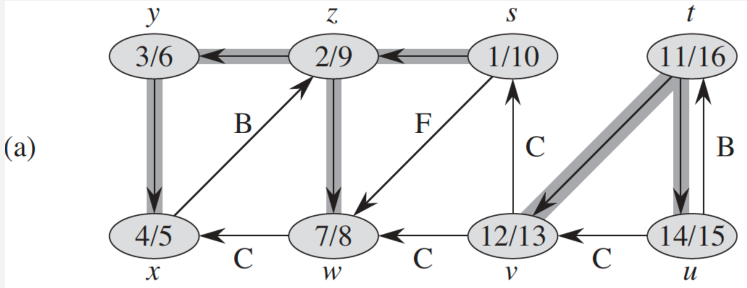
- In the execution of DFS-VISIT(G, v)
  - $u = v.\pi$ if and only if DFS-VISIT(G, v) was called during a search of u's adjacency list.
    - Example, x has established its predecessor y.
  - Vertex v is a ***descendant*** of vertex u in the depth-first forest if and only if v is discovered during the time in which u is grey.
    - Example, x is a ***descendant*** of vertex y.
- When we first explore and edge (u, v), if the color of vertex v is WHITE, edge (u, v) is a ***tree edge***.

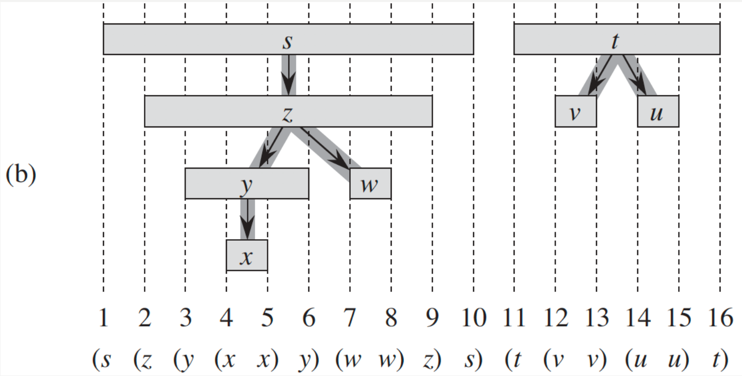




THE PROPERTY OF DFS #2

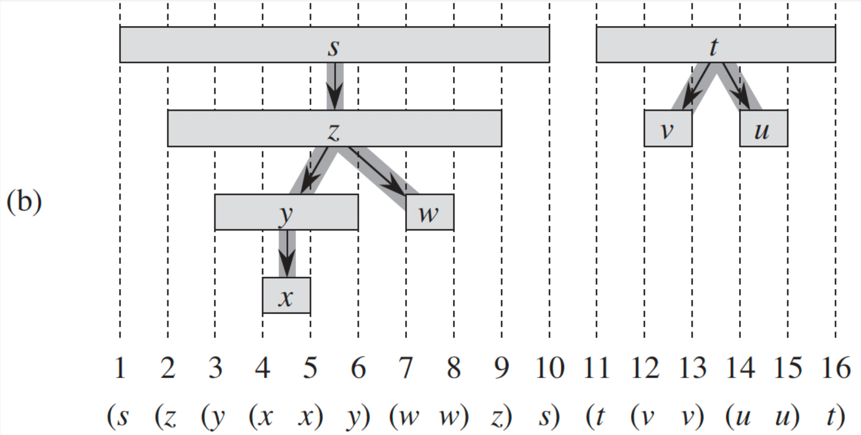
- The ***parenthesis structure***:
  - Represent the discovery of vertex u with a left parenthesis "$(u$" and
  - Represent its finishing by a right parenthesis "$u)$"
  - Then the history of discoveries and finishings makes a well-formed expression in the sense that the parentheses are properly nested.





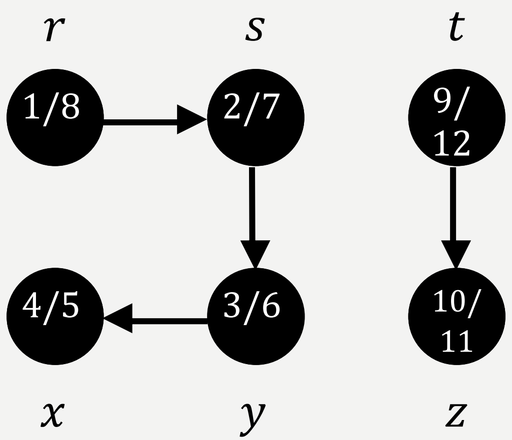
THE PROPERTY OF DFS #2
THEOREM 22.7

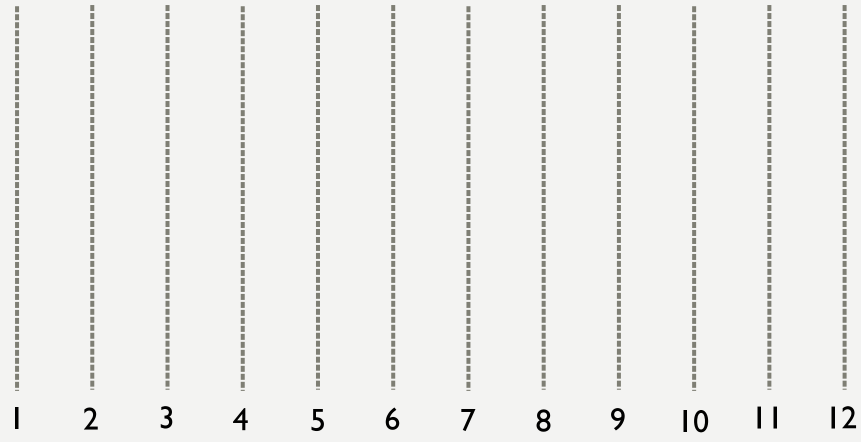
- **Parenthesis theorem**
  - In any depth-first search of a (directed or undirected) graph $G = (V, E)$, for any two vertices u and v, exactly one of the following three conditions holds:
    1. the intervals $[u.d, u.f]$ and $[v.d, v.f]$ are entirely disjoint, and neither u or v is a descendant of the other in the depth first forest,
    2. the interval $[u.d, u.f]$ is contained entirely within the interval $[v.d, v.f]$, and u is a descendant of v in a depth-first tree, or
    3. the interval $[v.d, v.f]$ is contained entirely within the interval $[u.d, u.f]$, and v is a descendant of u in a depth-first tree.



THE PROPERTY OF DFS #2
PRACTICE

- Consider the depth-first forest (left) we developed as the result of running DFS.
- Show the intervals of the discovery and finish of each vertex on the diagram (right).
- Show the corresponding parenthesization.





NEXT UP
DEPTH-FIRST SEARCH

REFERENCE

- Screenshots are taken from the textbook.