# Decomposition of Graphs: Previsit and Postvisit Orders

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

## Graph Algorithms
## Data Structures and Algorithms

# Learning Objectives

- Compute the preorder and postorder numbers for a DFS.
- Understand why these numbers might be important.

# Outline

# Need to Record Data

- Plain `DFS` just marks all vertices as visited.
- Need to keep track of other data to be useful.
- Augment functions to store additional information.

# Previsit and Postvisit Functions

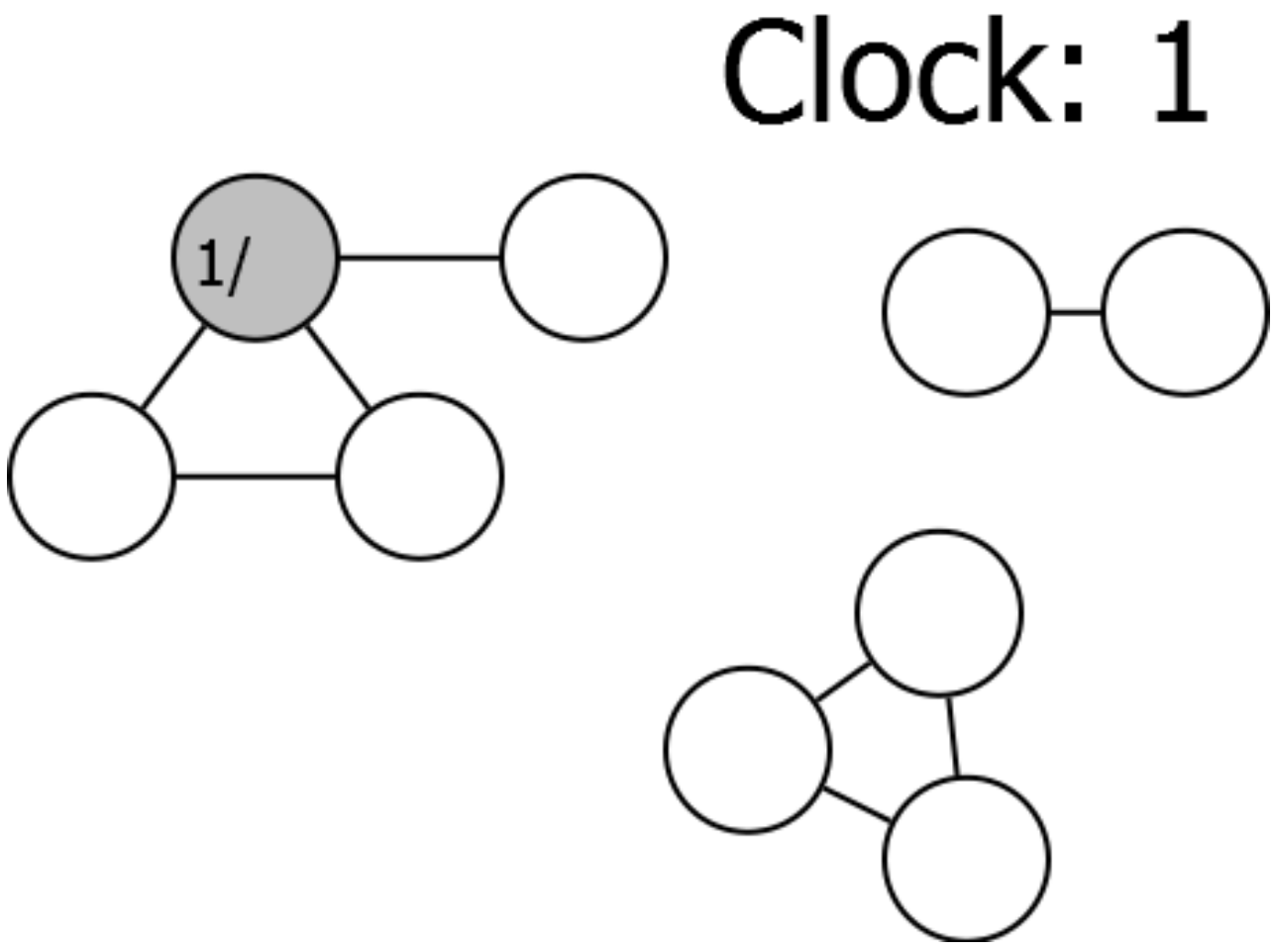
## Explore($v$)

```
visited(v) ← true
previsit(v)
for (v, w) ∈ E:
    if not visited(w):
        explore(w)
postvisit(v)
```

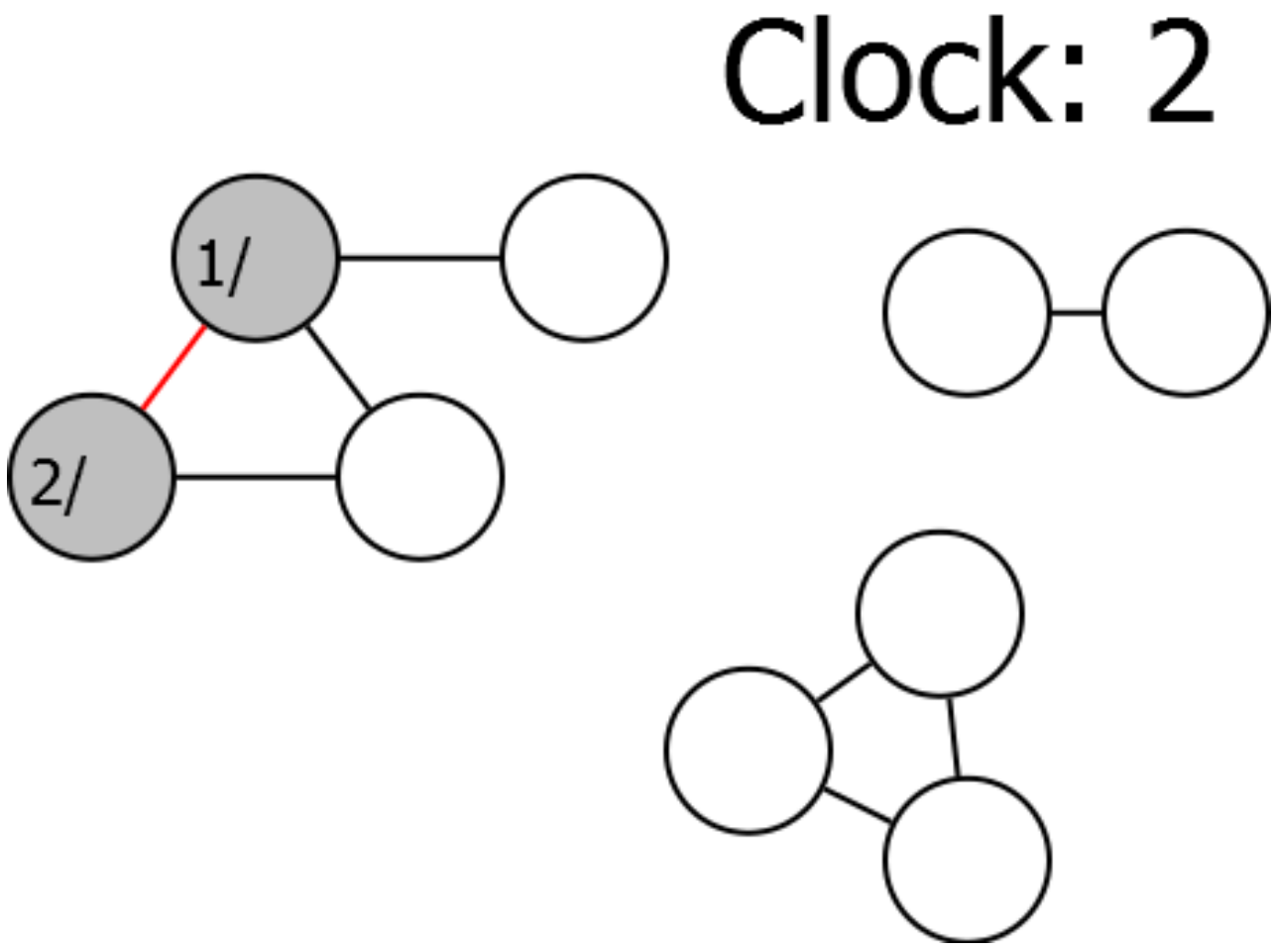
# Clock

- Keep track of order of visits.
- Clock ticks at each pre-/post- visit.
- Records previsit and postvisit times for each $v$.

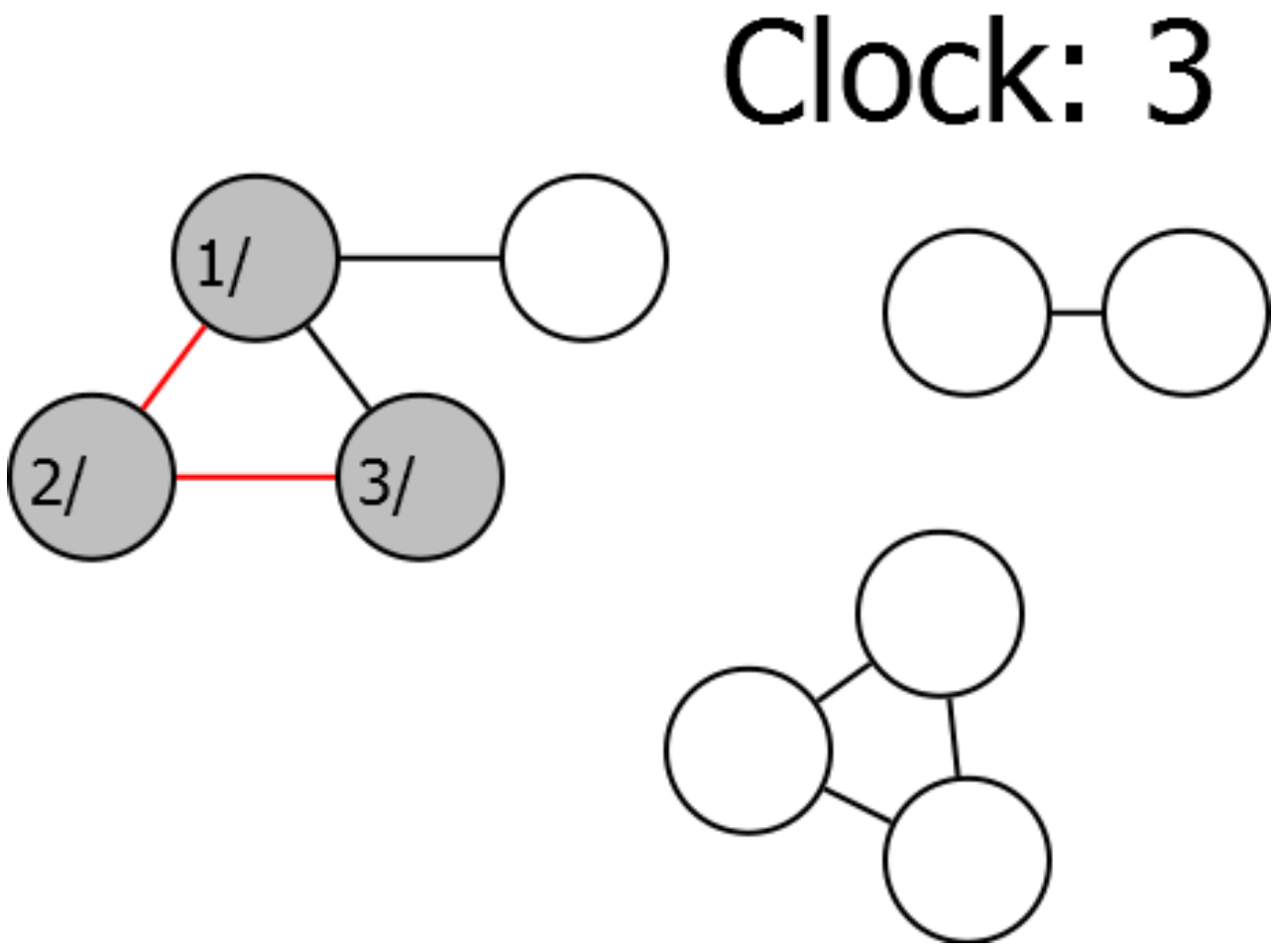
# Example



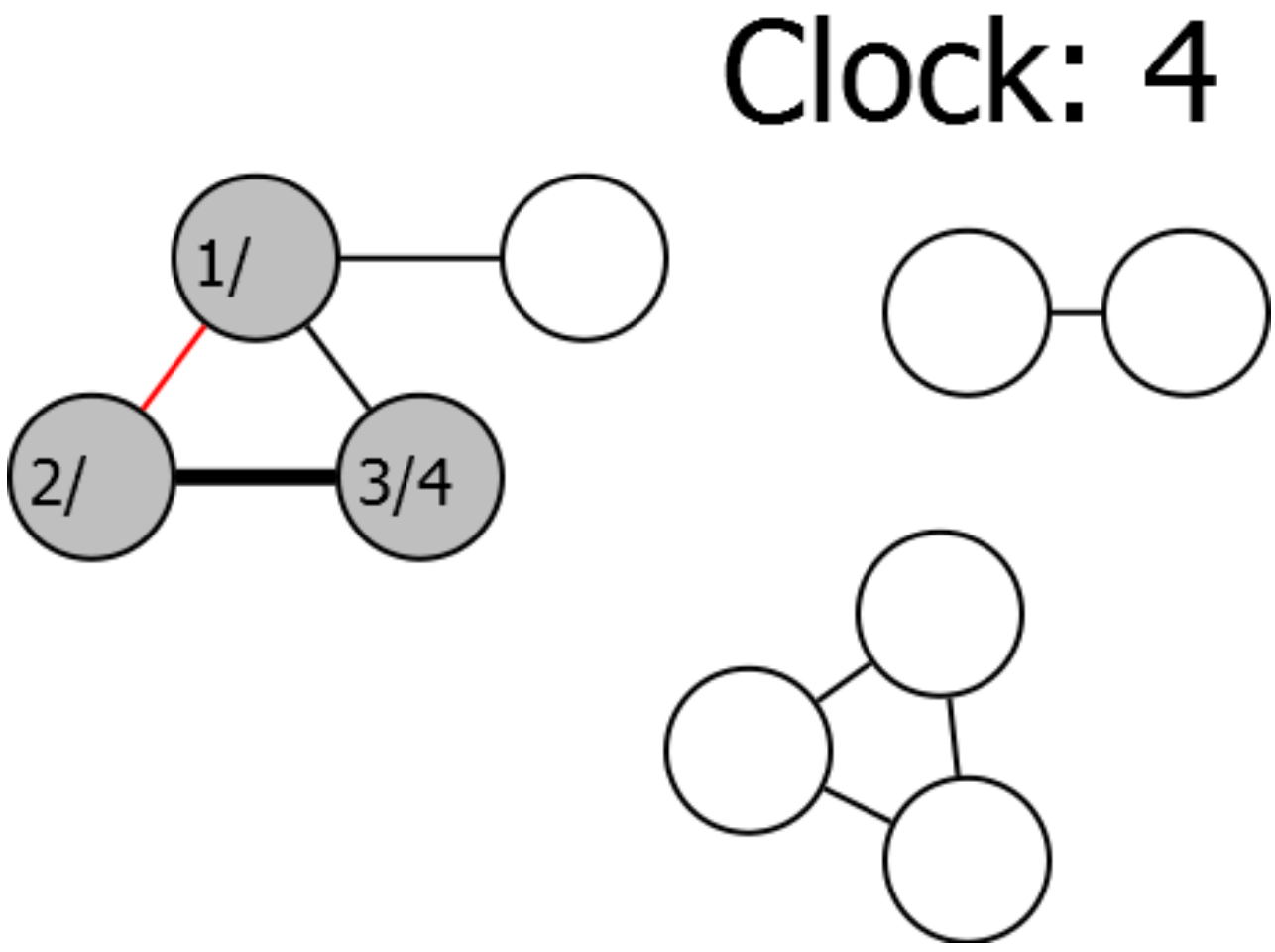
# Example



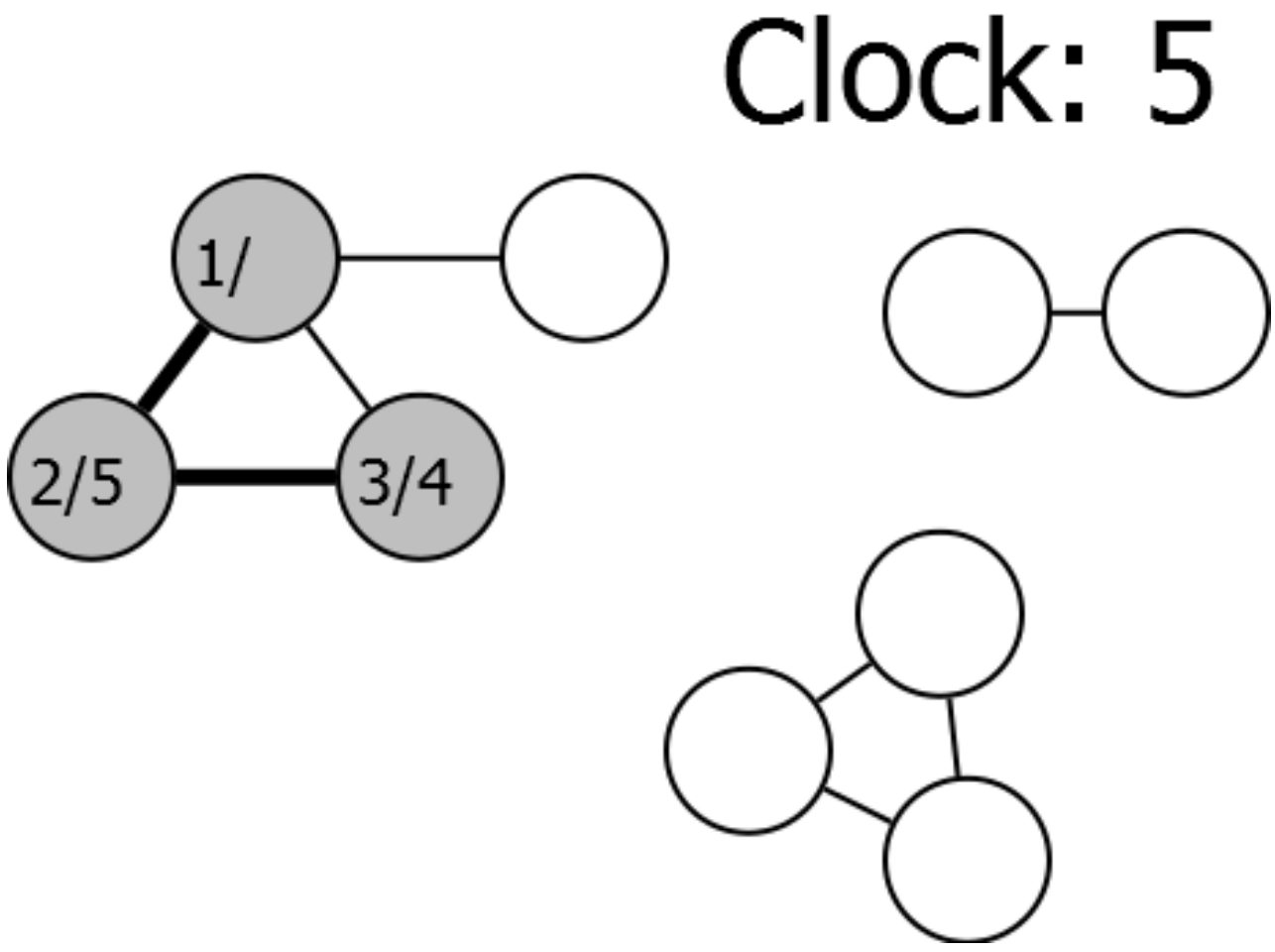
# Example



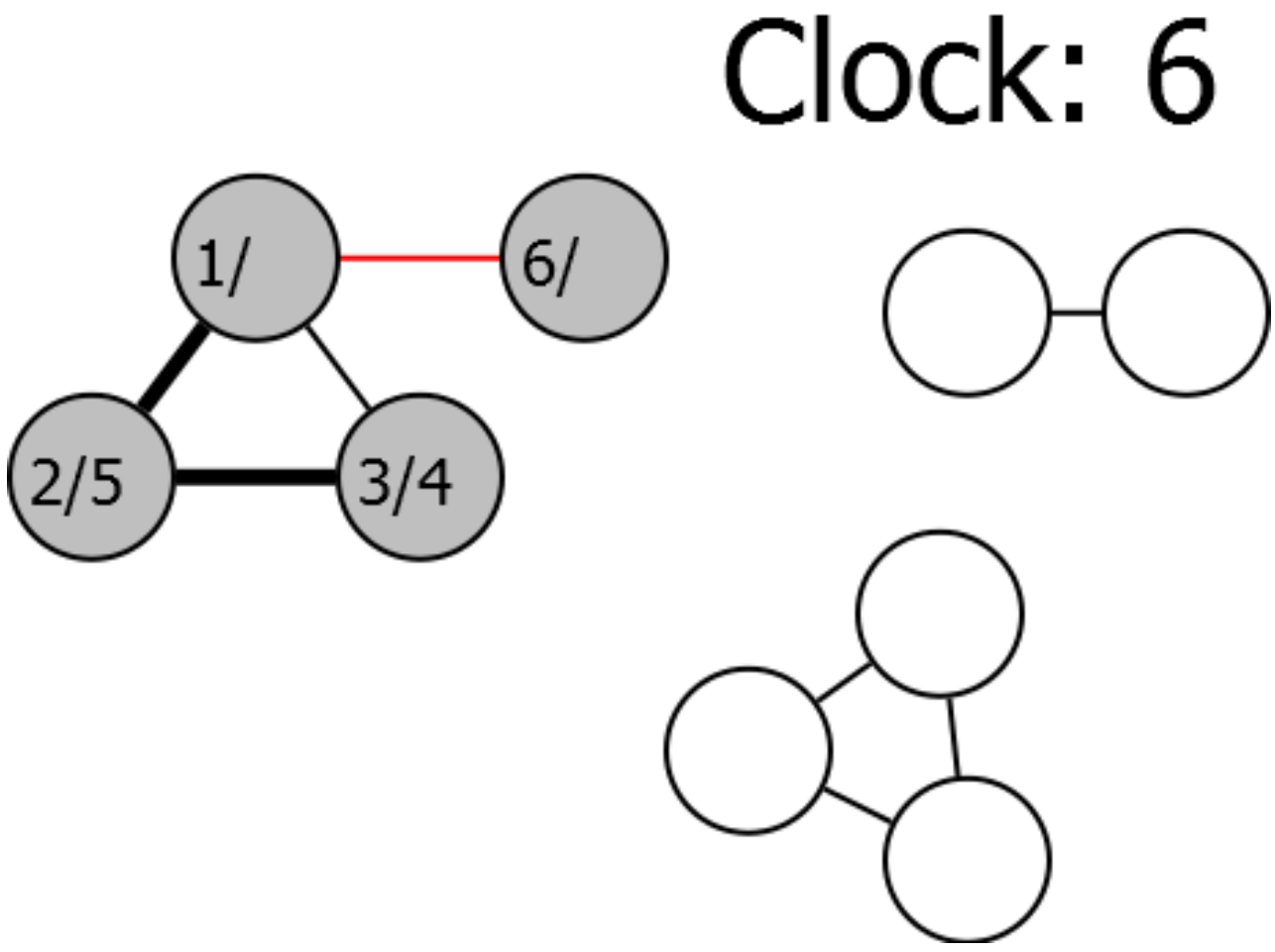
# Example



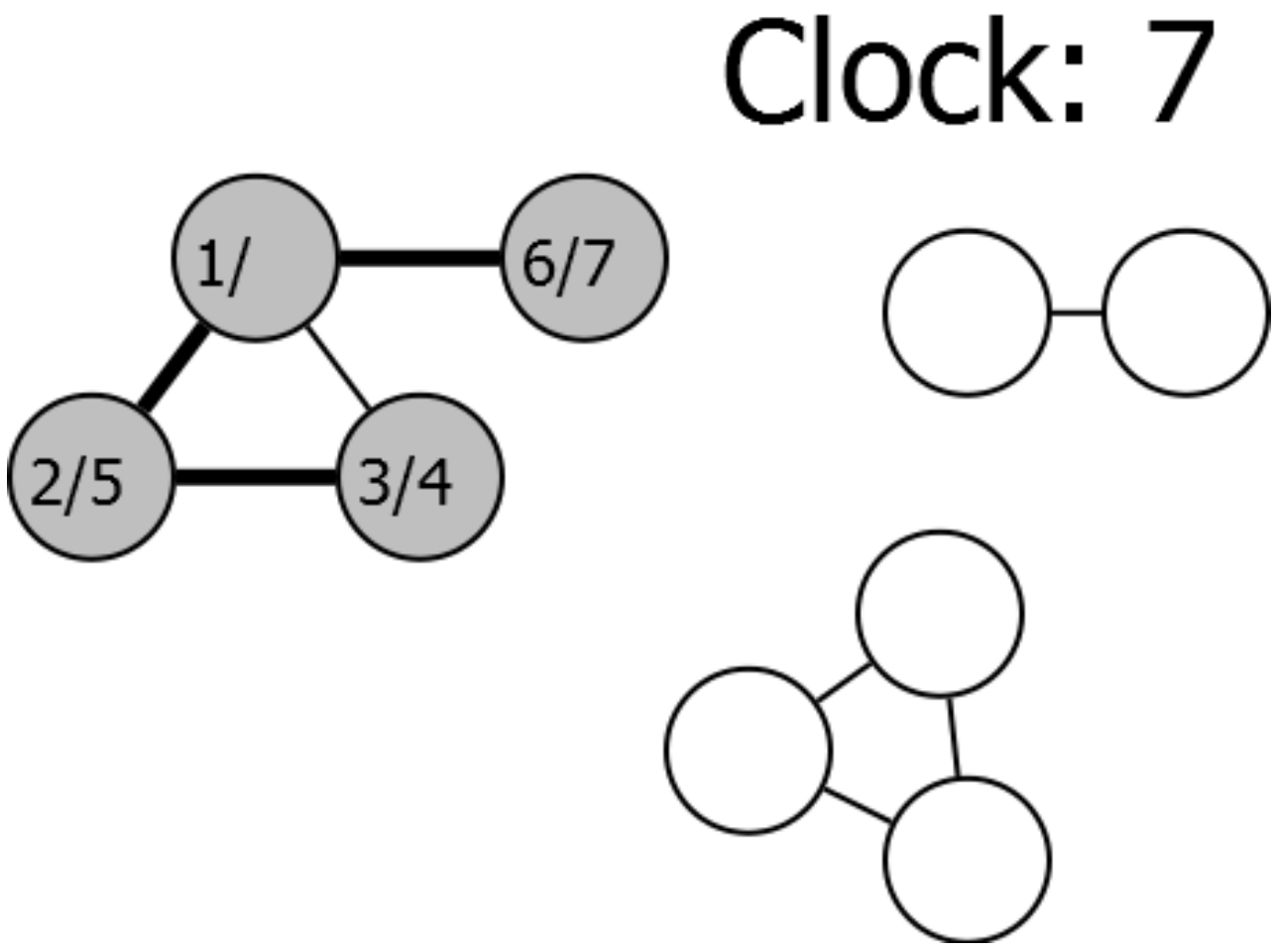
# Example



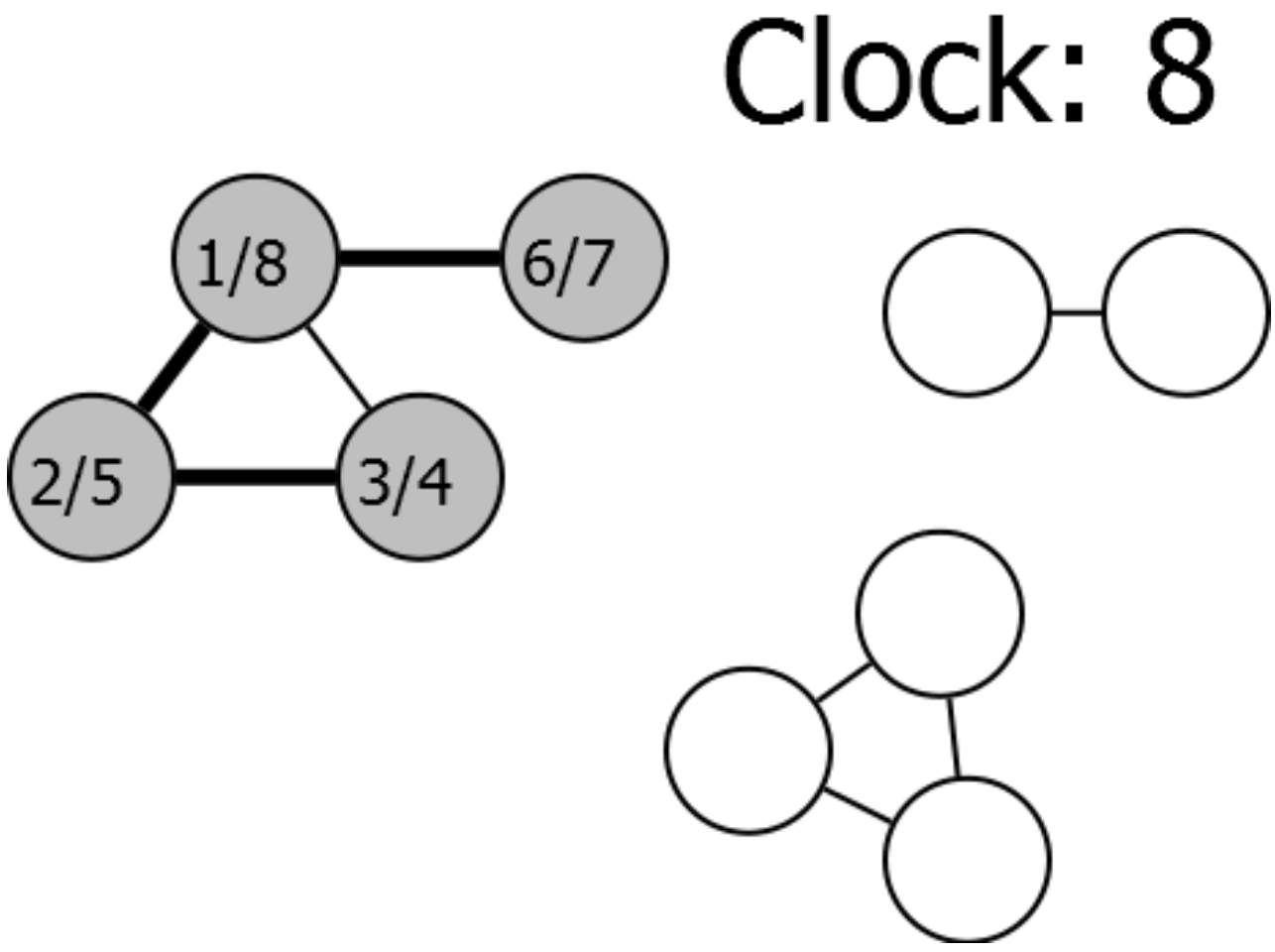
# Example



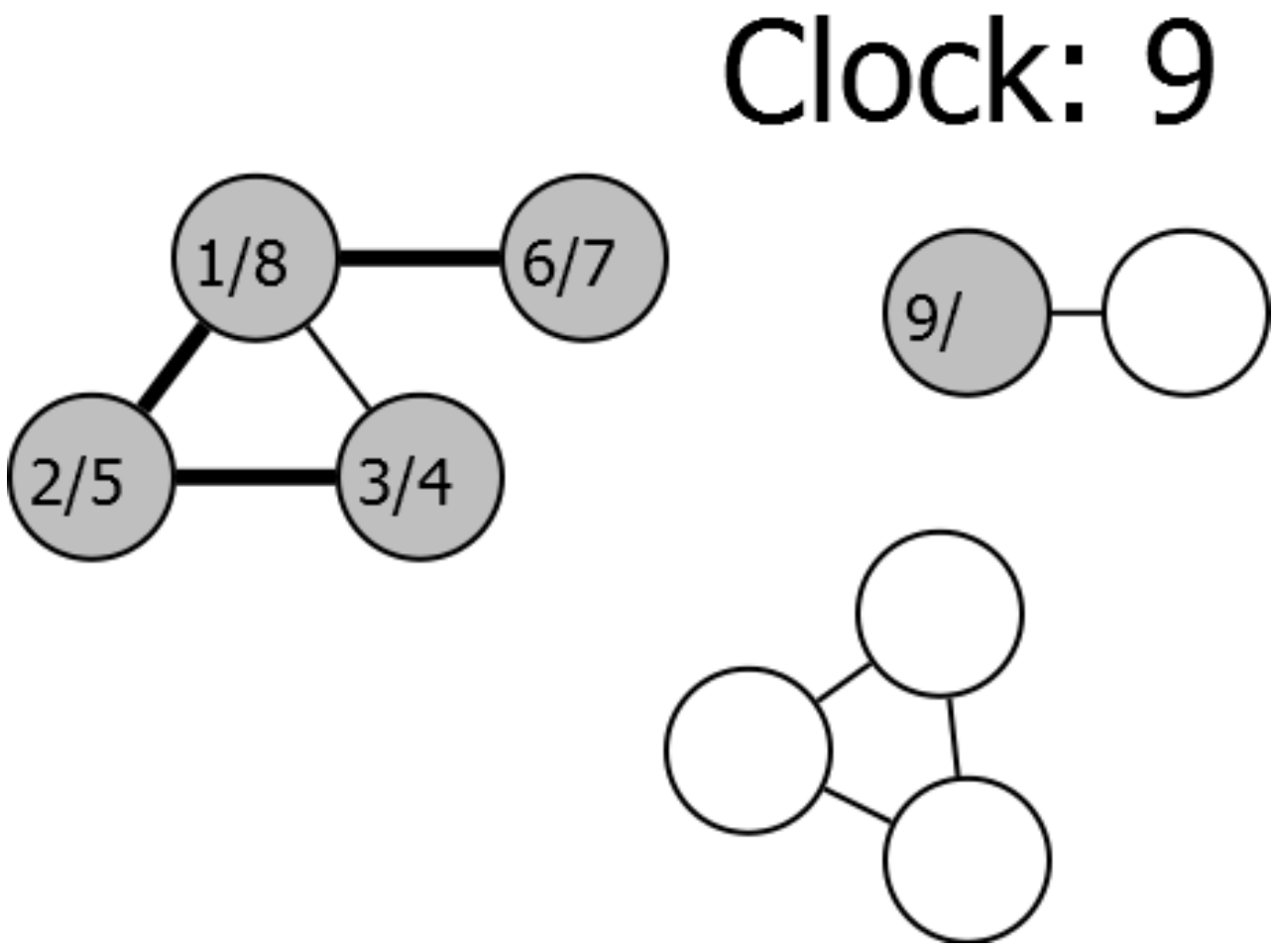
# Example



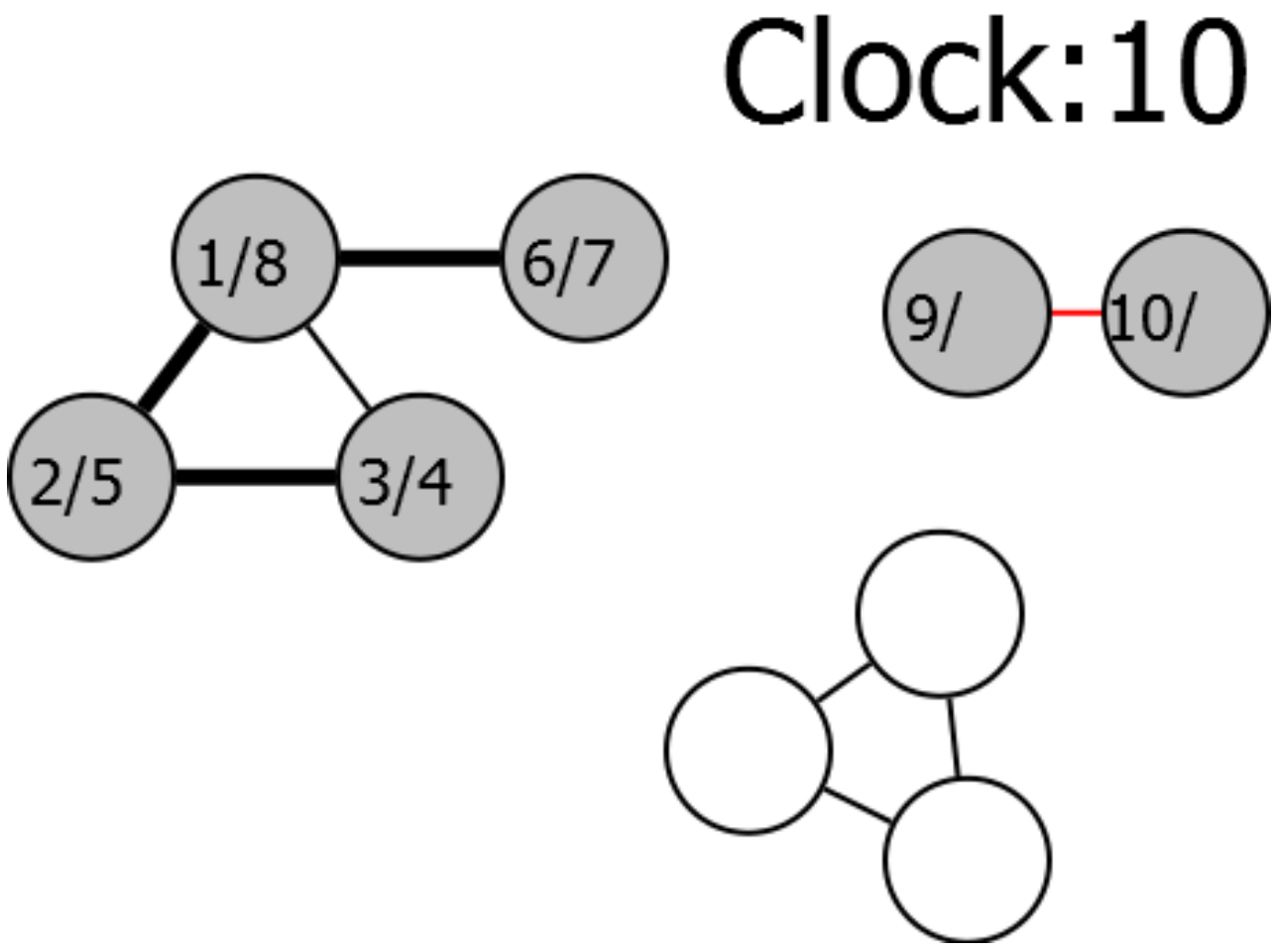
# Example



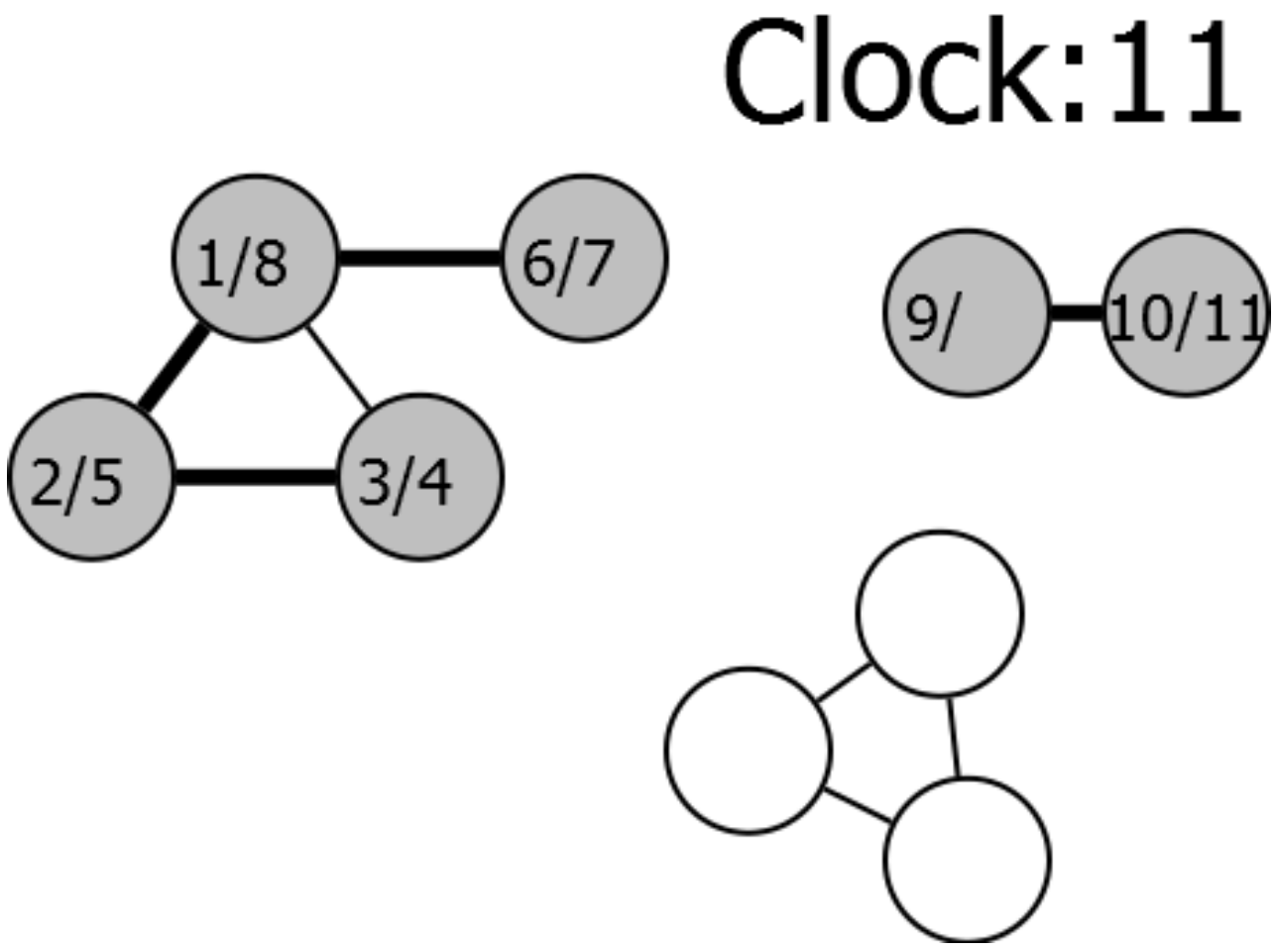
# Example



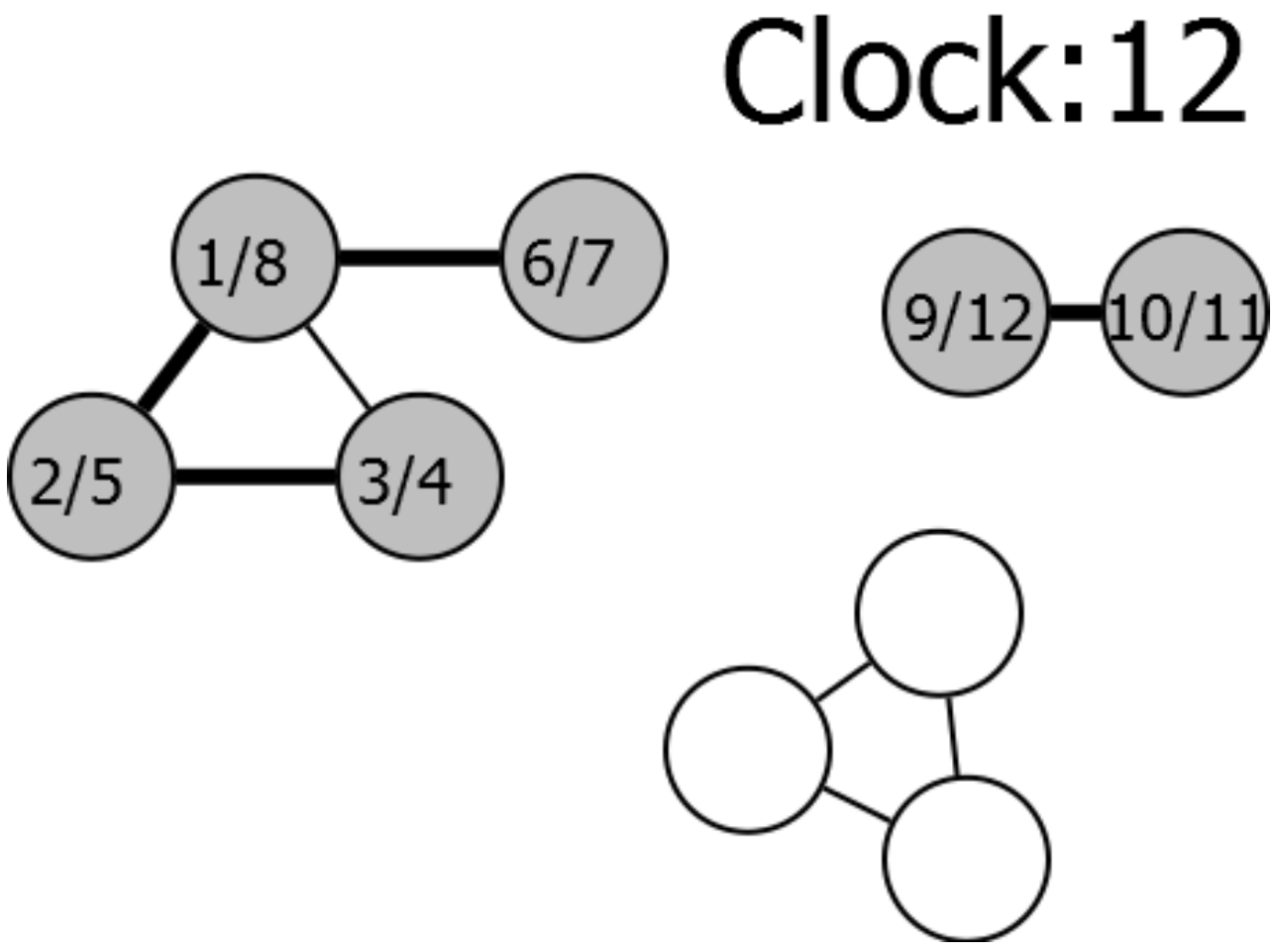
# Example



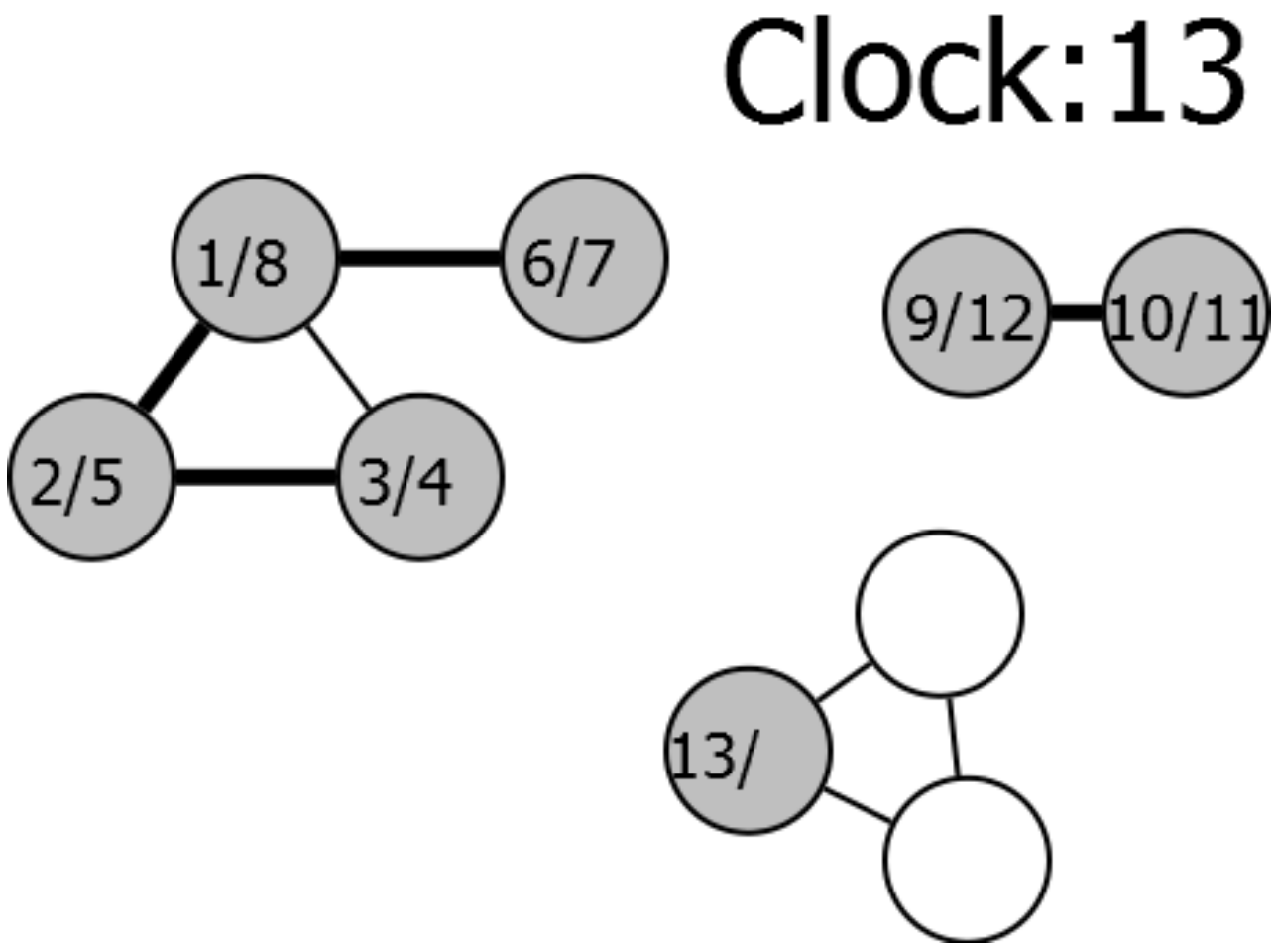
# Example



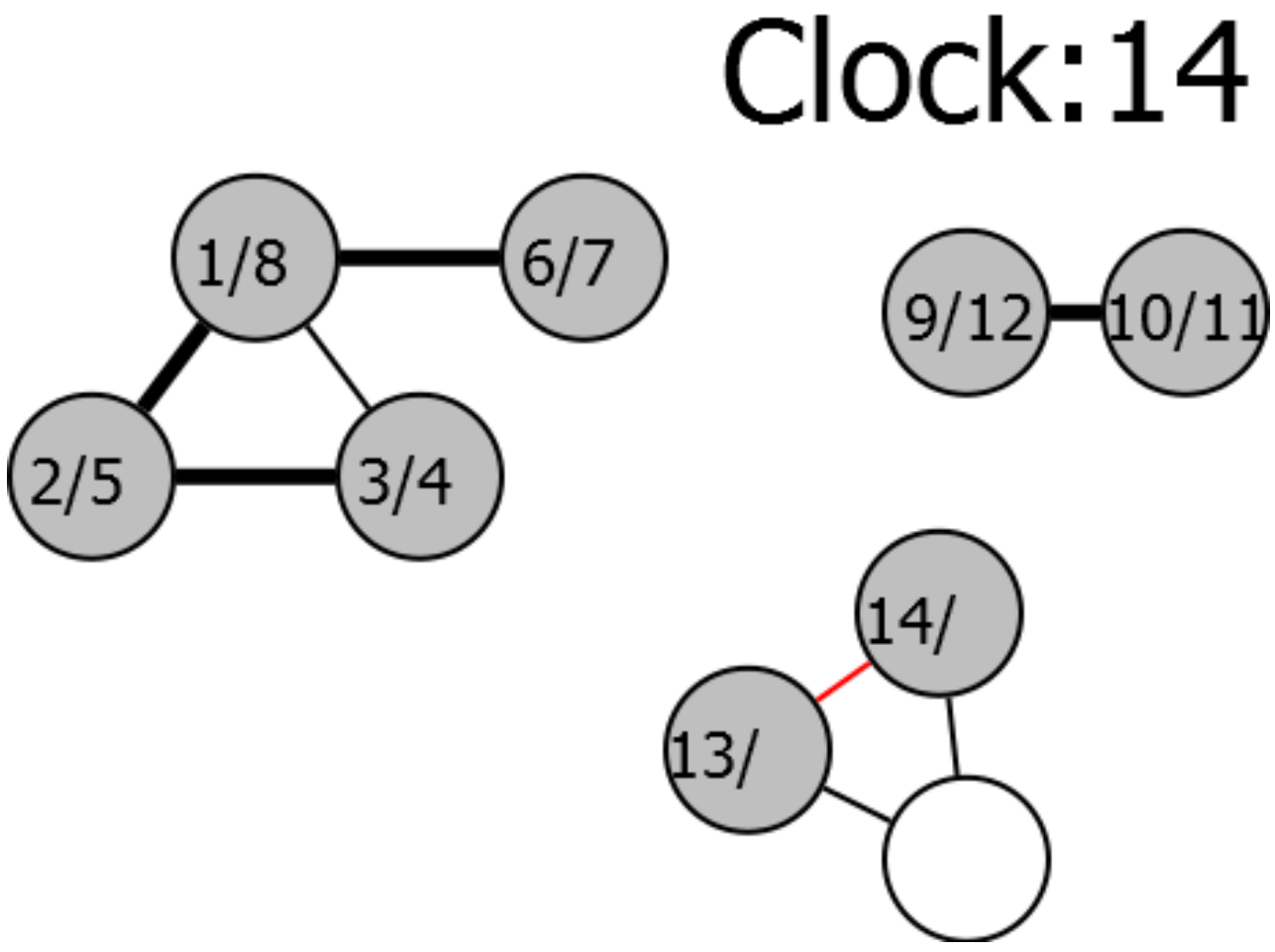
# Example



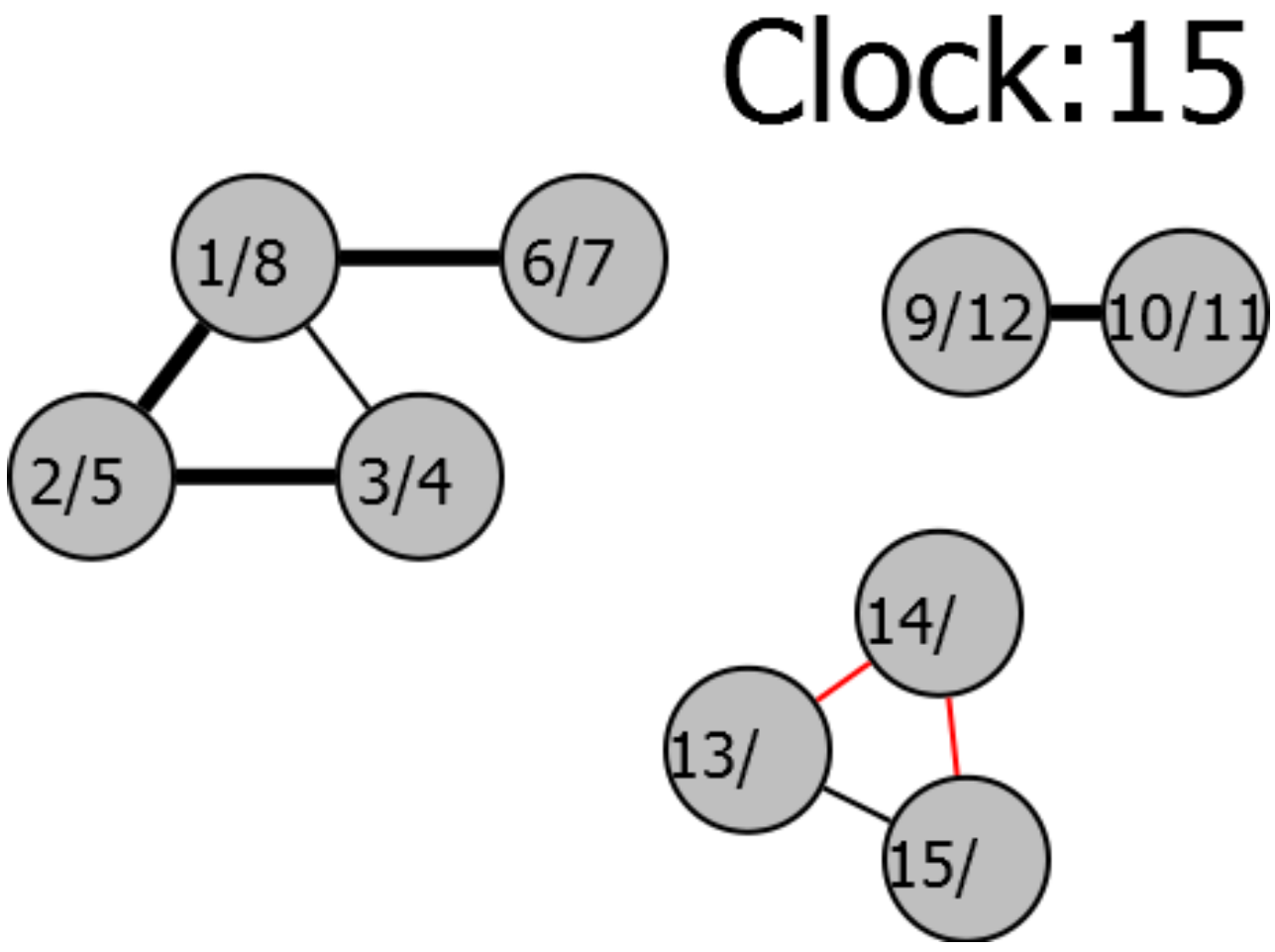
# Example



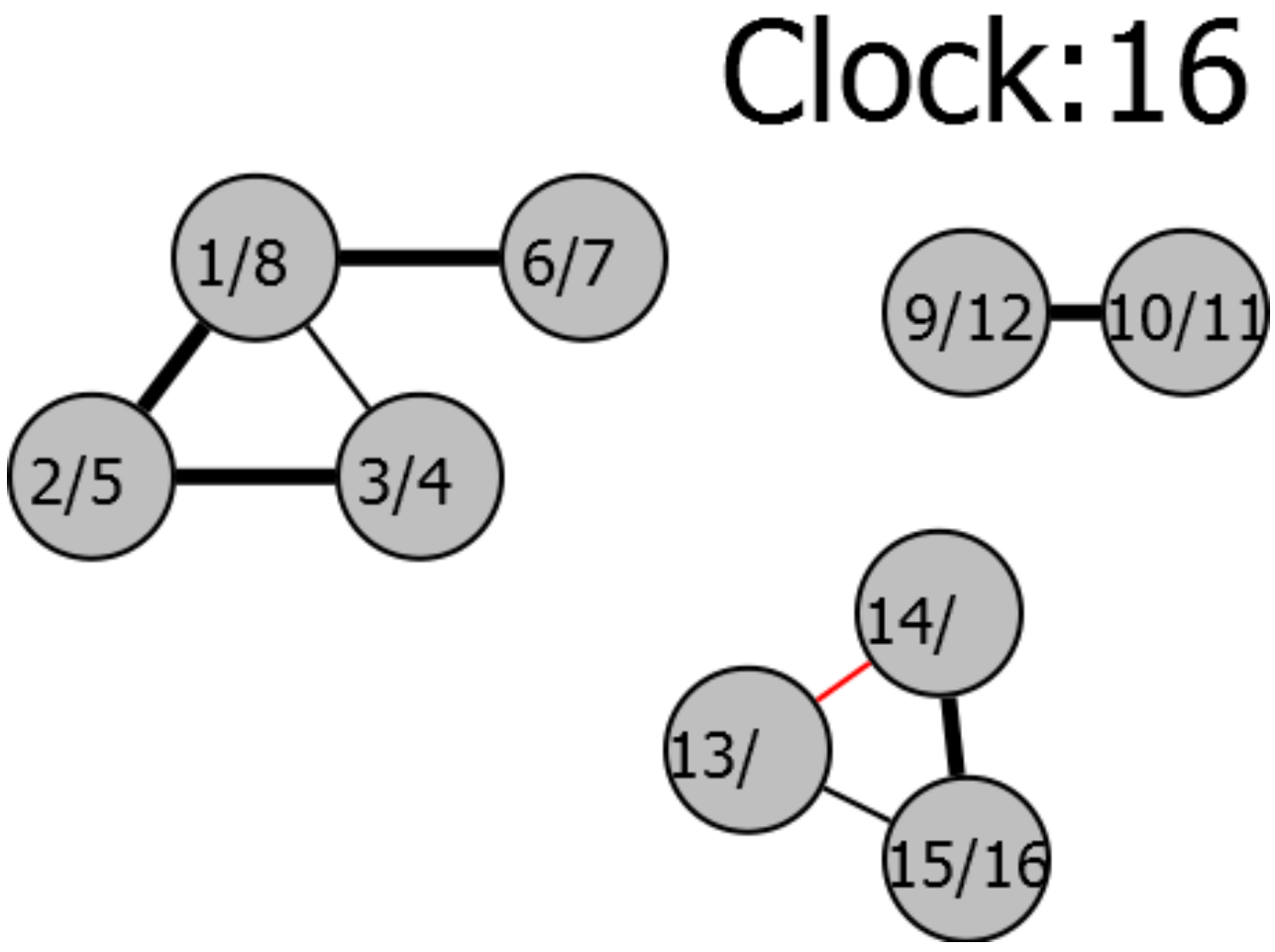
# Example



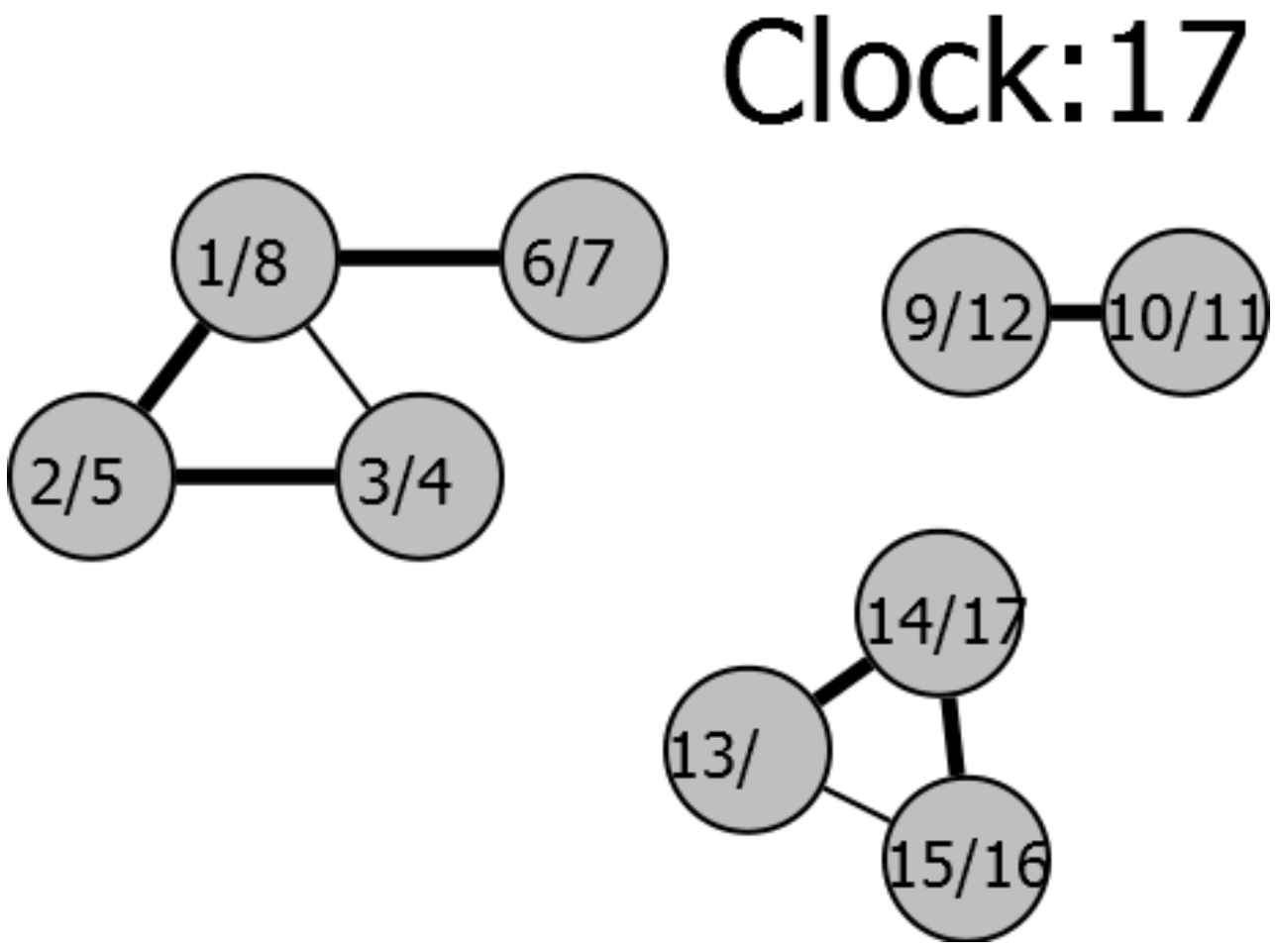
# Example



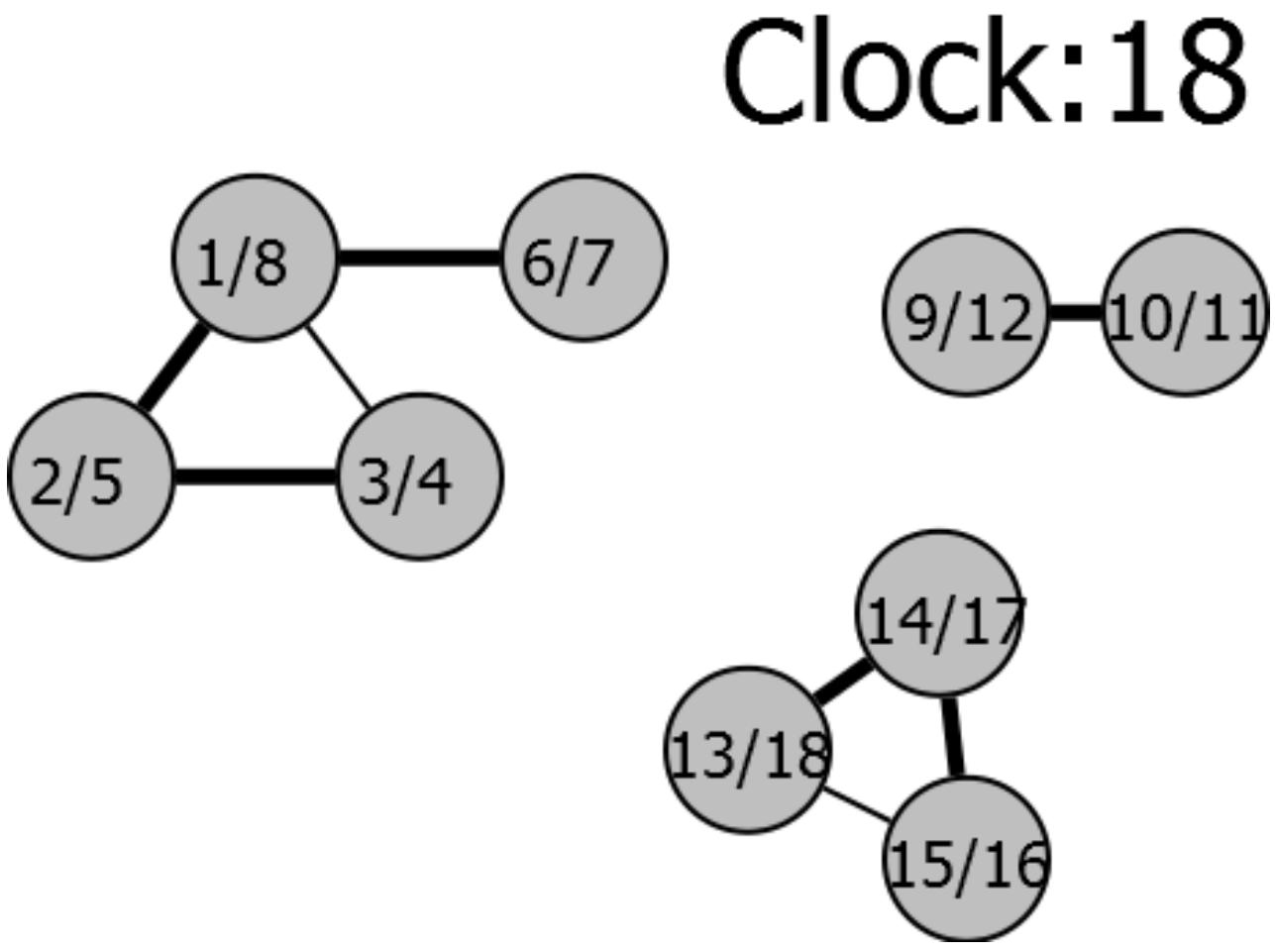
# Example



# Example



# Example



# Computing Pre- and Post- Numbers

Initialize `clock` to 1.

**previsit($v$)**

$\texttt{pre}(v) \leftarrow \texttt{clock}$

$\texttt{clock} \leftarrow \texttt{clock} + 1$

**postvisit($v$)**

$\texttt{post}(v) \leftarrow \texttt{clock}$

$\texttt{clock} \leftarrow \texttt{clock} + 1$

# Outline

# Result

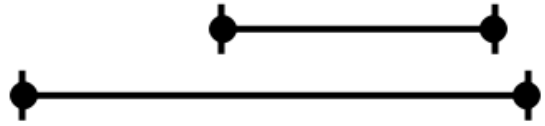
Previsit and Postvisit numbers tell us about the execution of `DFS`.

## Lemma

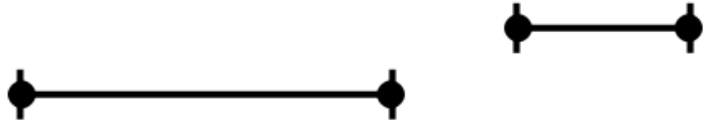
For any vertices $u, v$ the intervals $[\texttt{pre}(u), \texttt{post}(u)]$ and $[\texttt{pre}(v), \texttt{post}(v)]$ are either nested or disjoint.

# Explanation

Nested

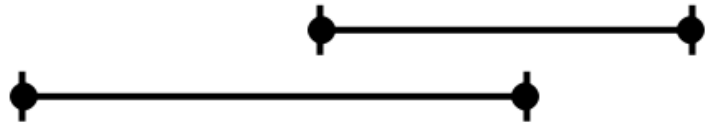


Disjoint



# Explanation

Interleaved (not possible)



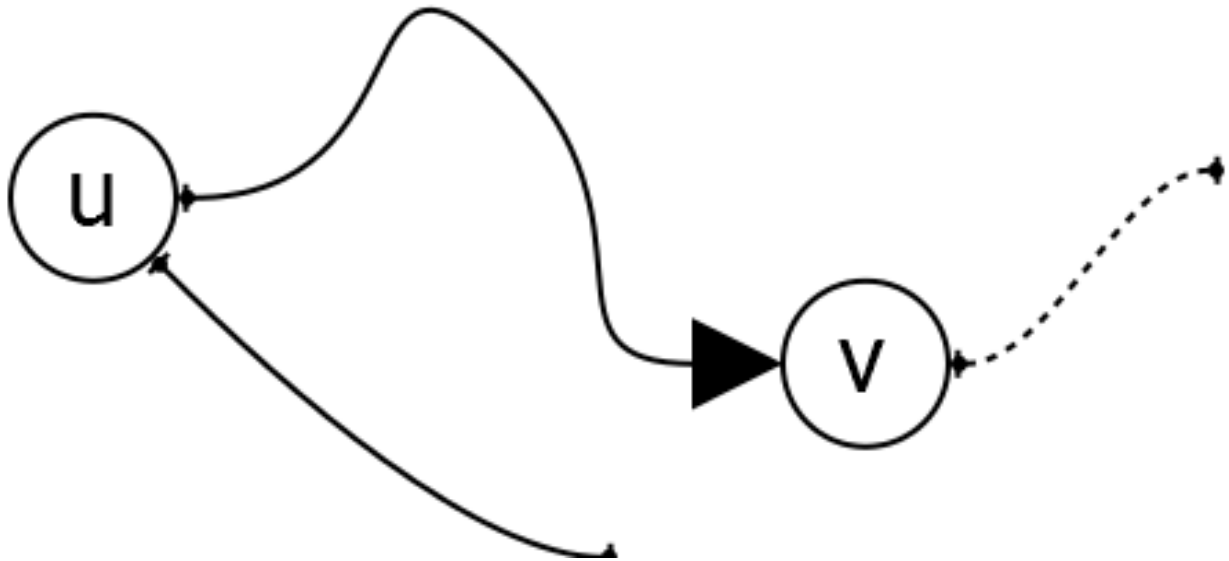
# Proof

Assume that $u$ visited before $v$. Two cases

- Find $v$ while exploring $u$ ($u$ an ancestor of $v$)
- Find $v$ after exploring $u$ ($u$ a cousin of $v$)

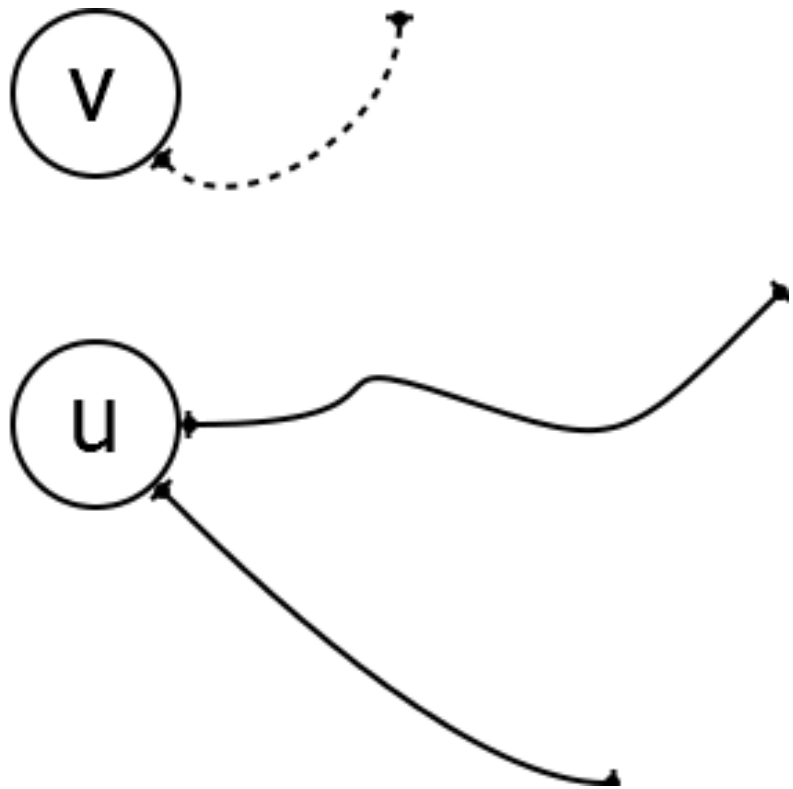
# Case I

If you explore $v$ while exploring $u$, cannot finish exploring $u$ until done exploring $v$. Therefore nested.



# Case II

If explore $v$ after finish exploring $u$, $\texttt{post}(u) < \texttt{pre}(v)$, therefore disjoint.



# Problem

Which of the following tables is not a valid set of pre- and post- orders?

| Vert. | Pre | Post |
|---|---|---|
| A | 1 | 8 |
| B | 9 | 10 |
| C | 3 | 4 |
| D | 2 | 7 |
| E | 5 | 6 |

| Vert. | Pre | Post |
|---|---|---|
| A | 1 | 9 |
| B | 8 | 10 |
| C | 2 | 7 |
| D | 3 | 6 |
| E | 4 | 5 |

# Solution

Which of the following tables is not a valid set of pre- and post- orders?

| Vert. | Pre | Post |
|---|---|---|
| A | 1 | 8 |
| B | 9 | 10 |
| C | 3 | 4 |
| D | 2 | 7 |
| E | 5 | 6 |

| Vert. | Pre | Post |
|---|---|---|
| A | 1 | 9 |
| B | 8 | 10 |
| C | 2 | 7 |
| D | 3 | 6 |
| E | 4 | 5 |

# Next Time

Directed graphs.