GRAPH

DEPTH-FIRST SEARCH

GRAPHS

01 Properties

02 Representation

03 Traversal algorithms

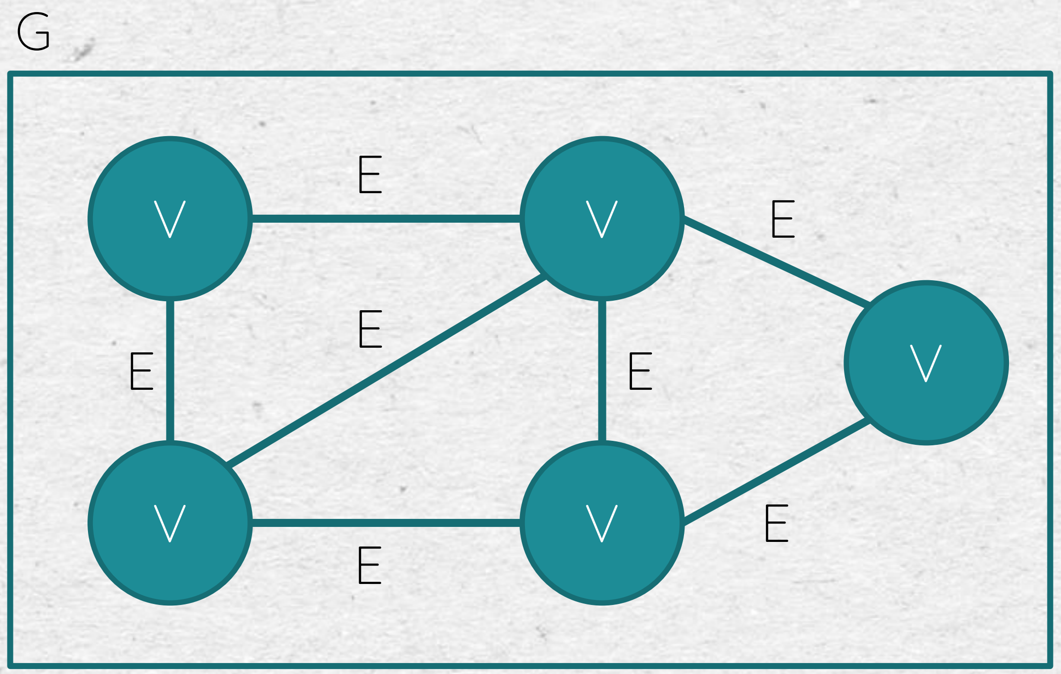
04 Pseudo-code

05 Implementation

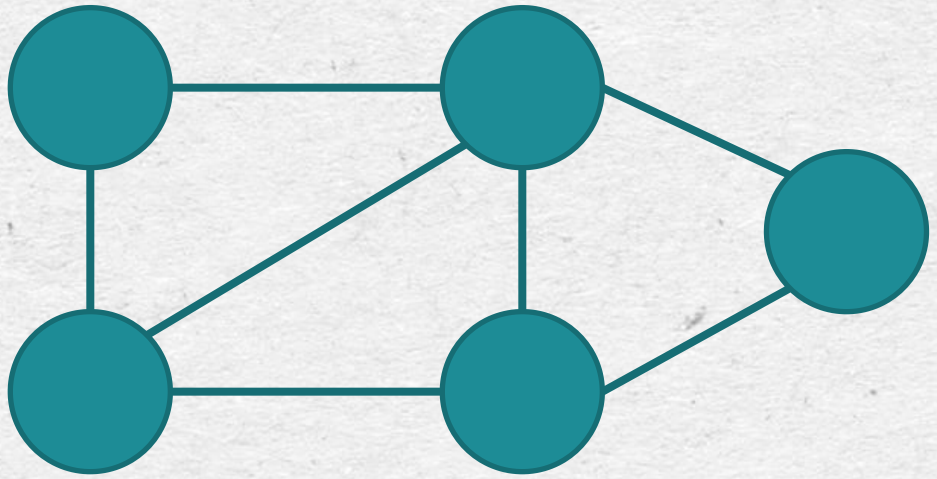
Graph

$$G = (V, E)$$

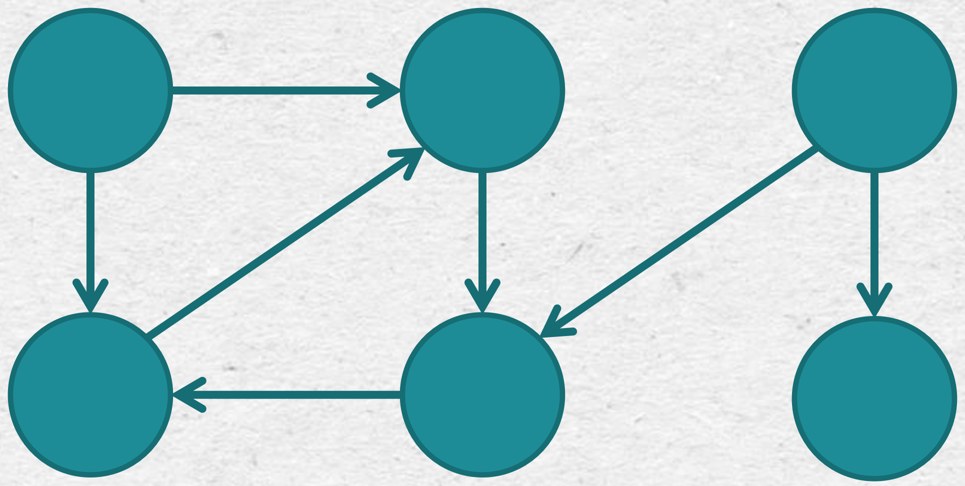
Graph
Vertex
Edge



Graph's Properties

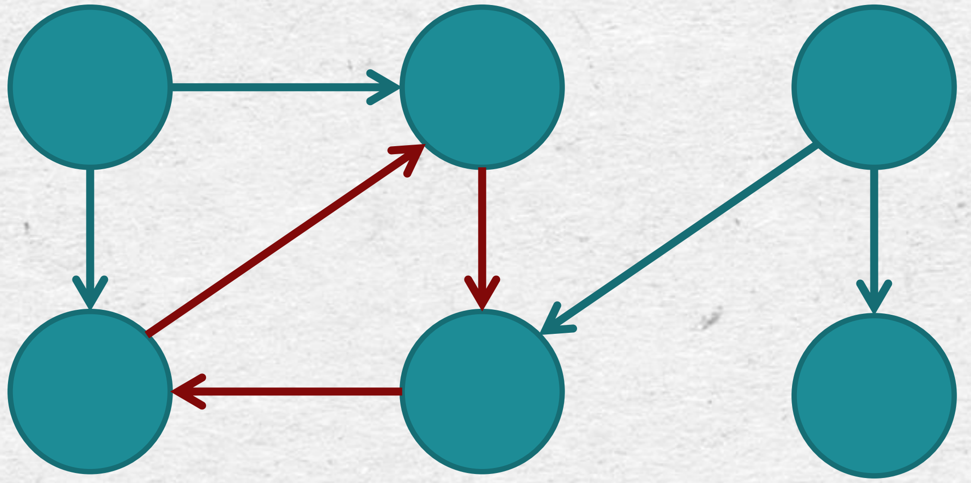


Undirected

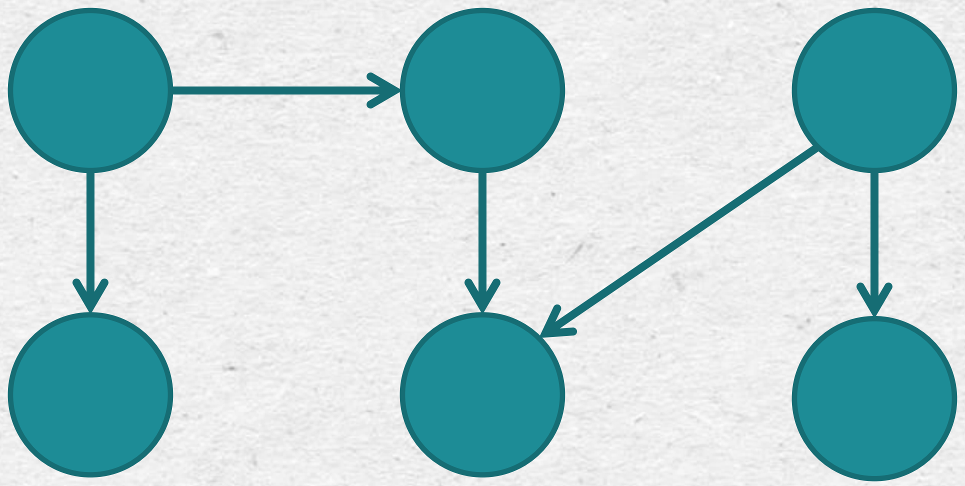


Directed

Graph's Properties

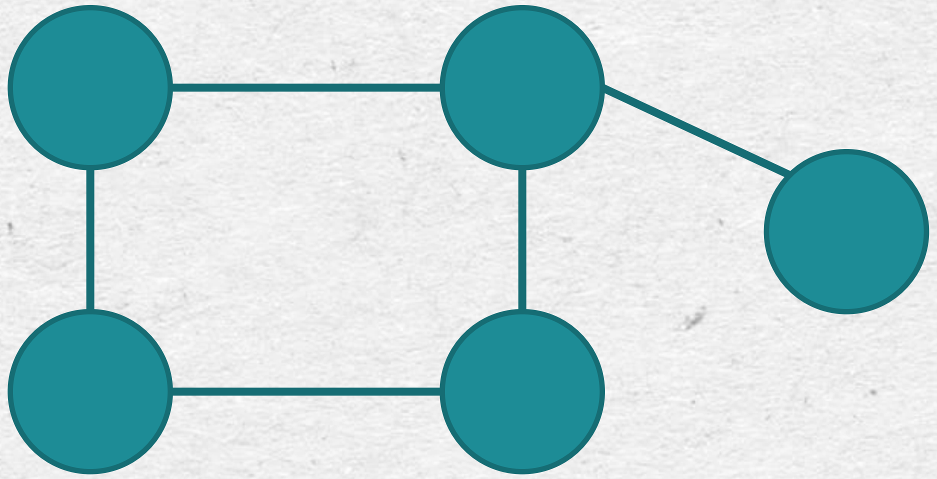


Cyclic

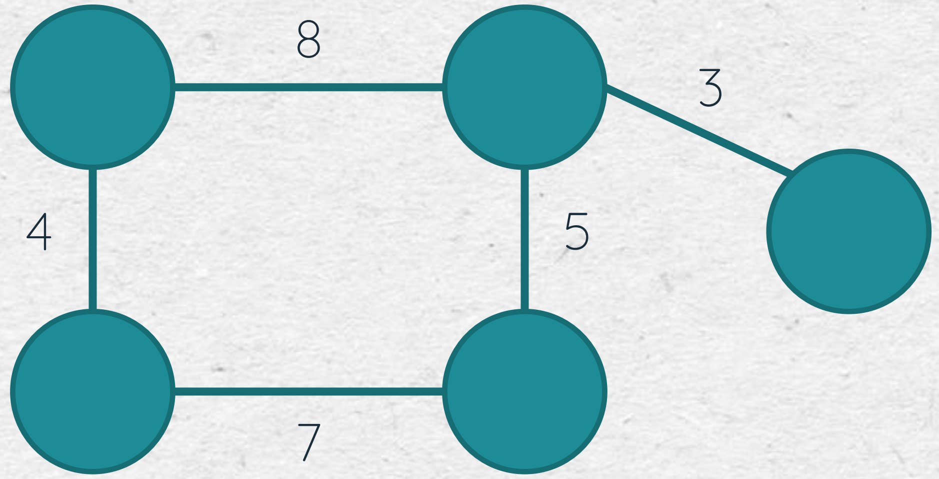


Acyclic

Graph's Properties

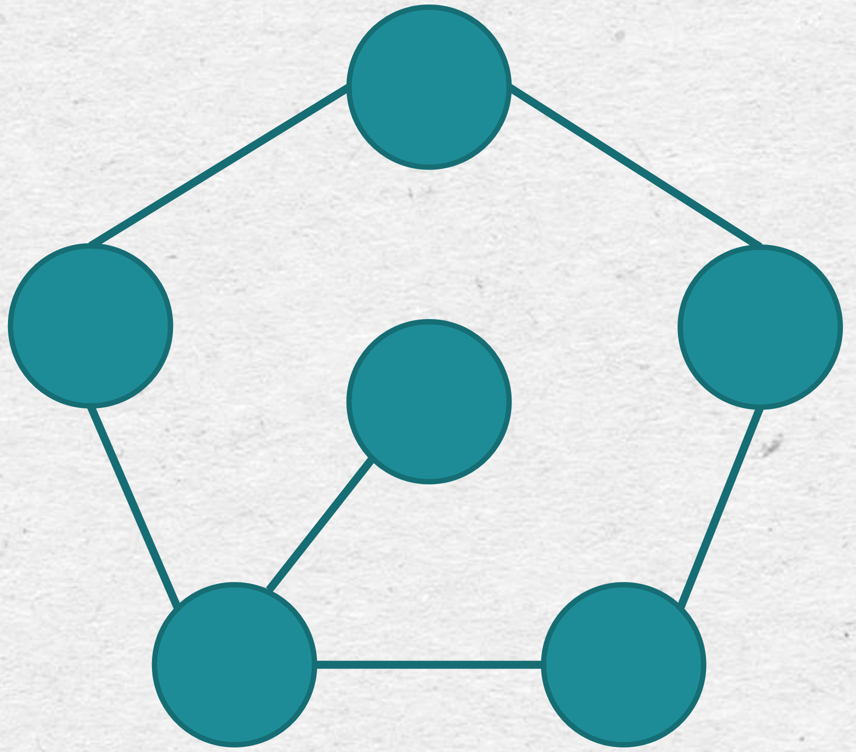


Unweighted

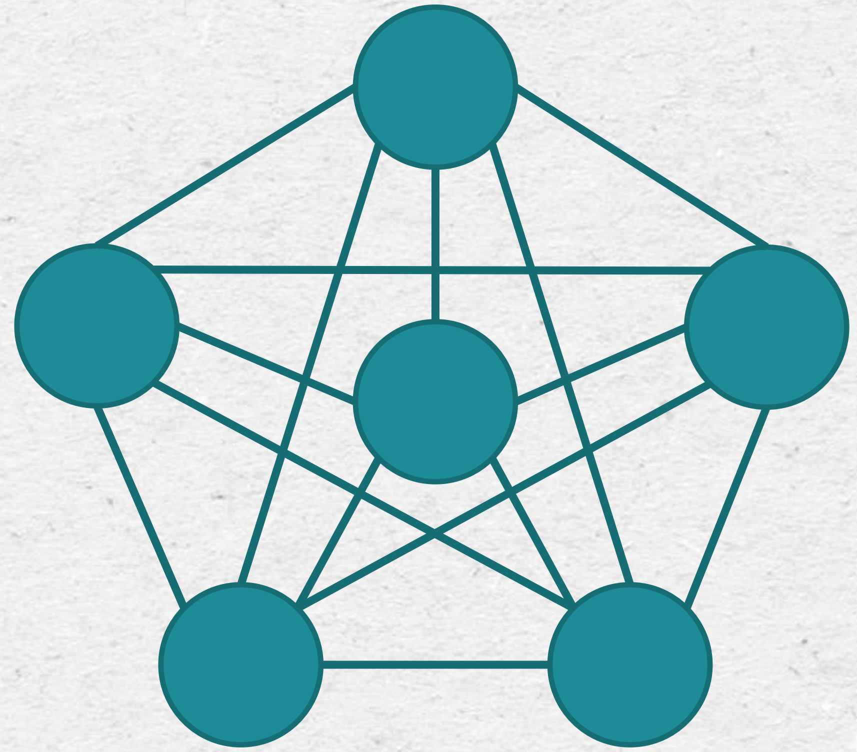


Weighted

Graph's Properties

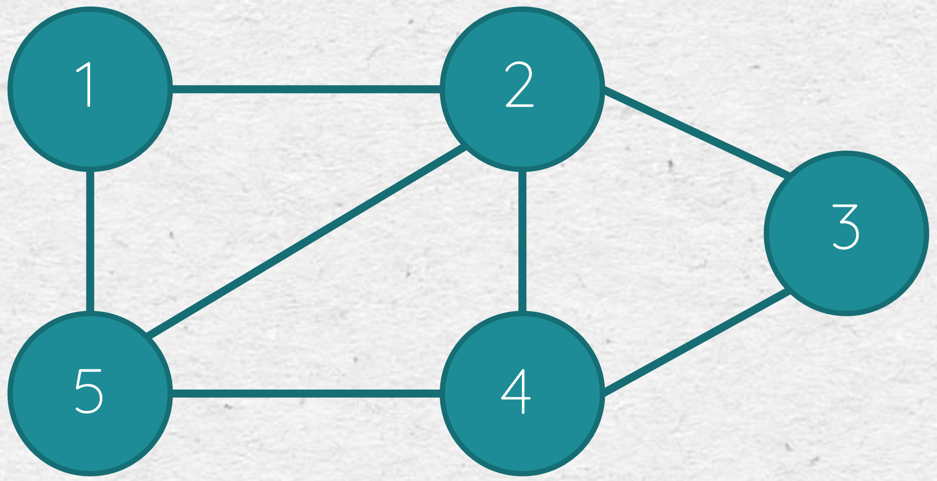


Sparse



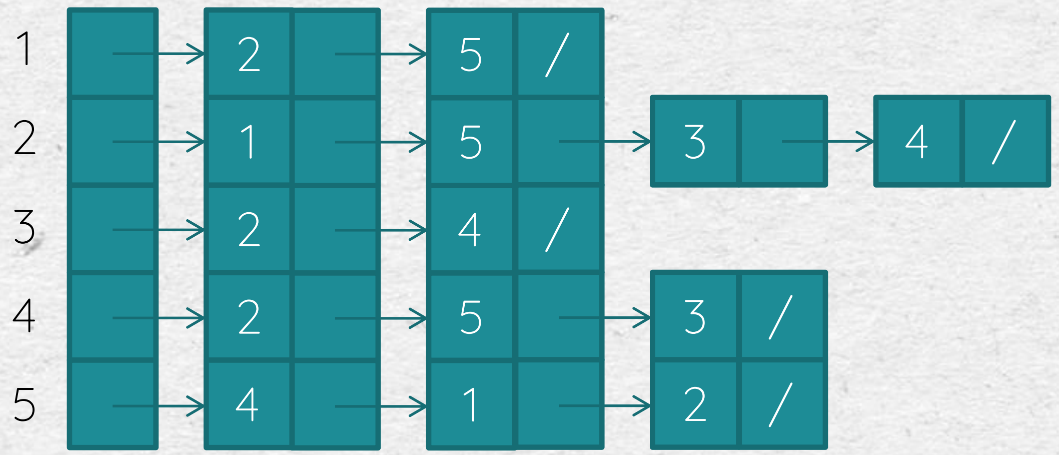
Dense

Representation



Undirected

Space



Adjacency-list

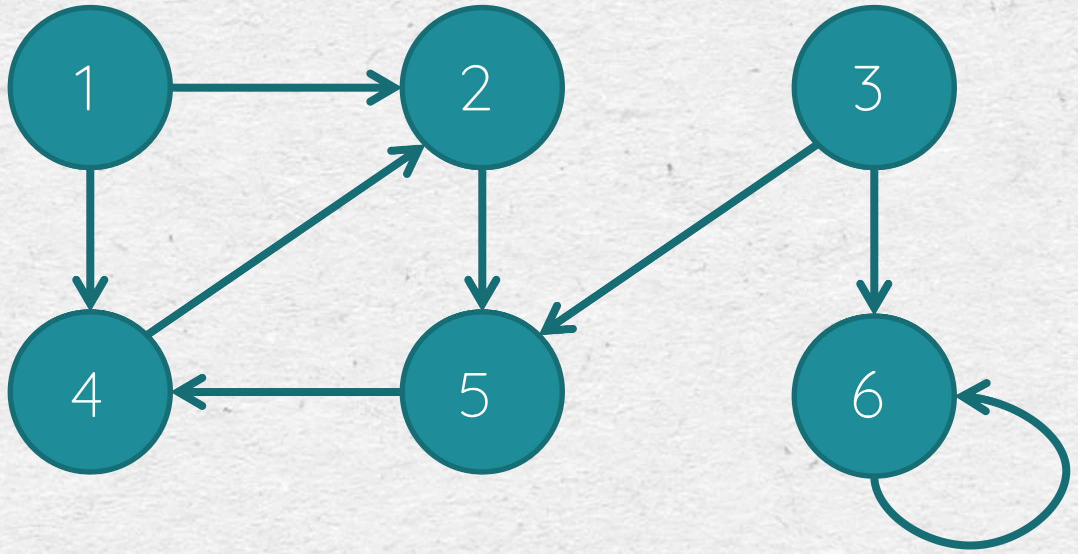
$\Theta(V + E)$

$$
\begin{array}{c|ccccc|}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 \\
2 & 1 & 0 & 1 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 \\
\hline
\end{array}
$$

Adjacency-matrix

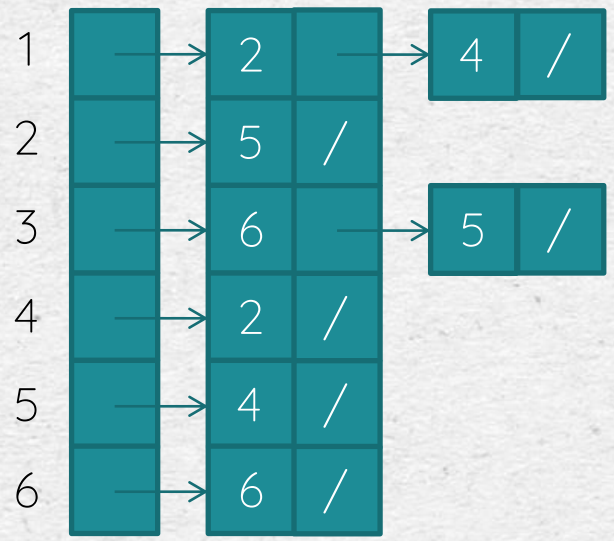
$\Theta(V^2)$

Representation



Undirected

Space



Adjacency-list

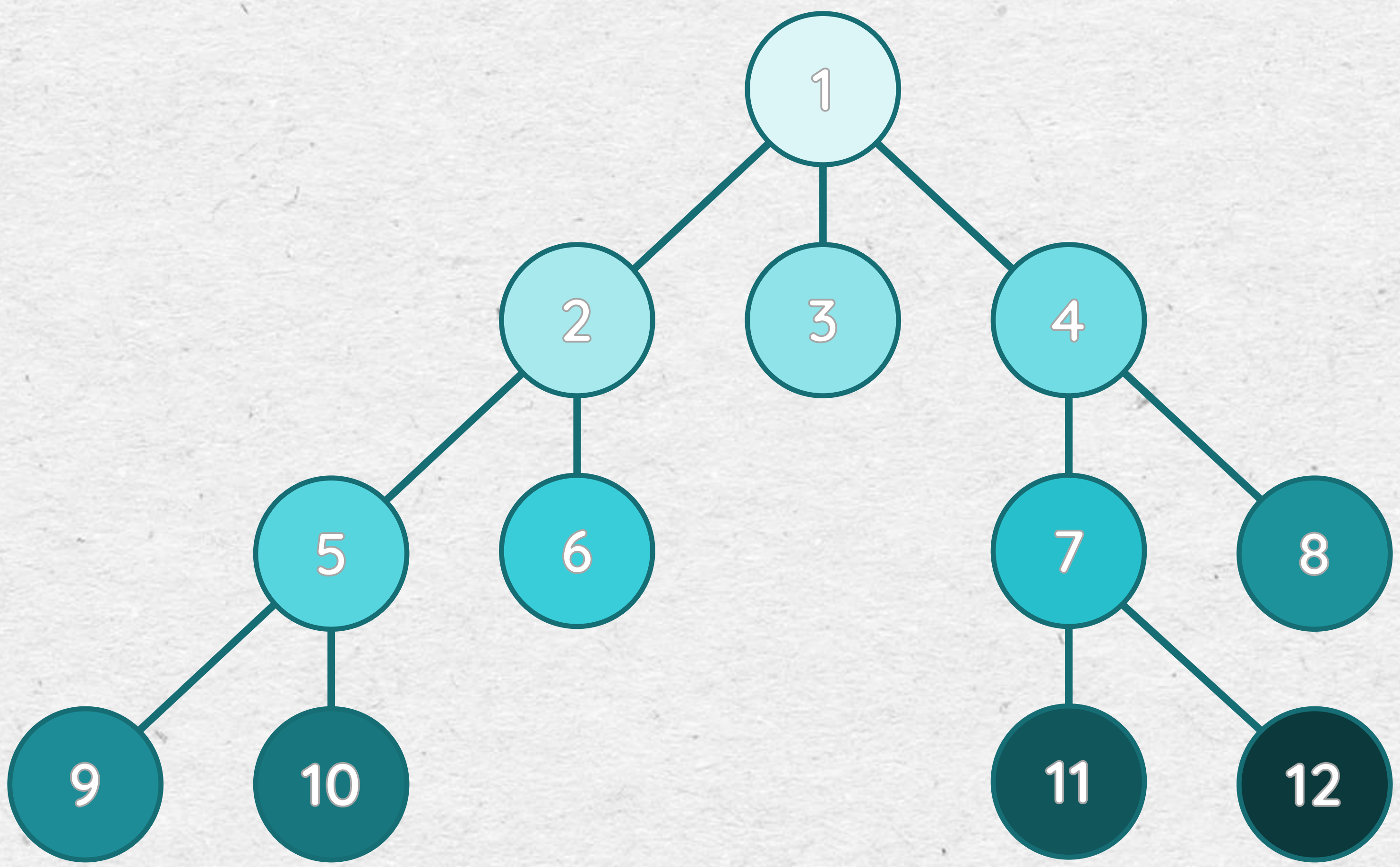
$\Theta(V, E)$

	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	6	5
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

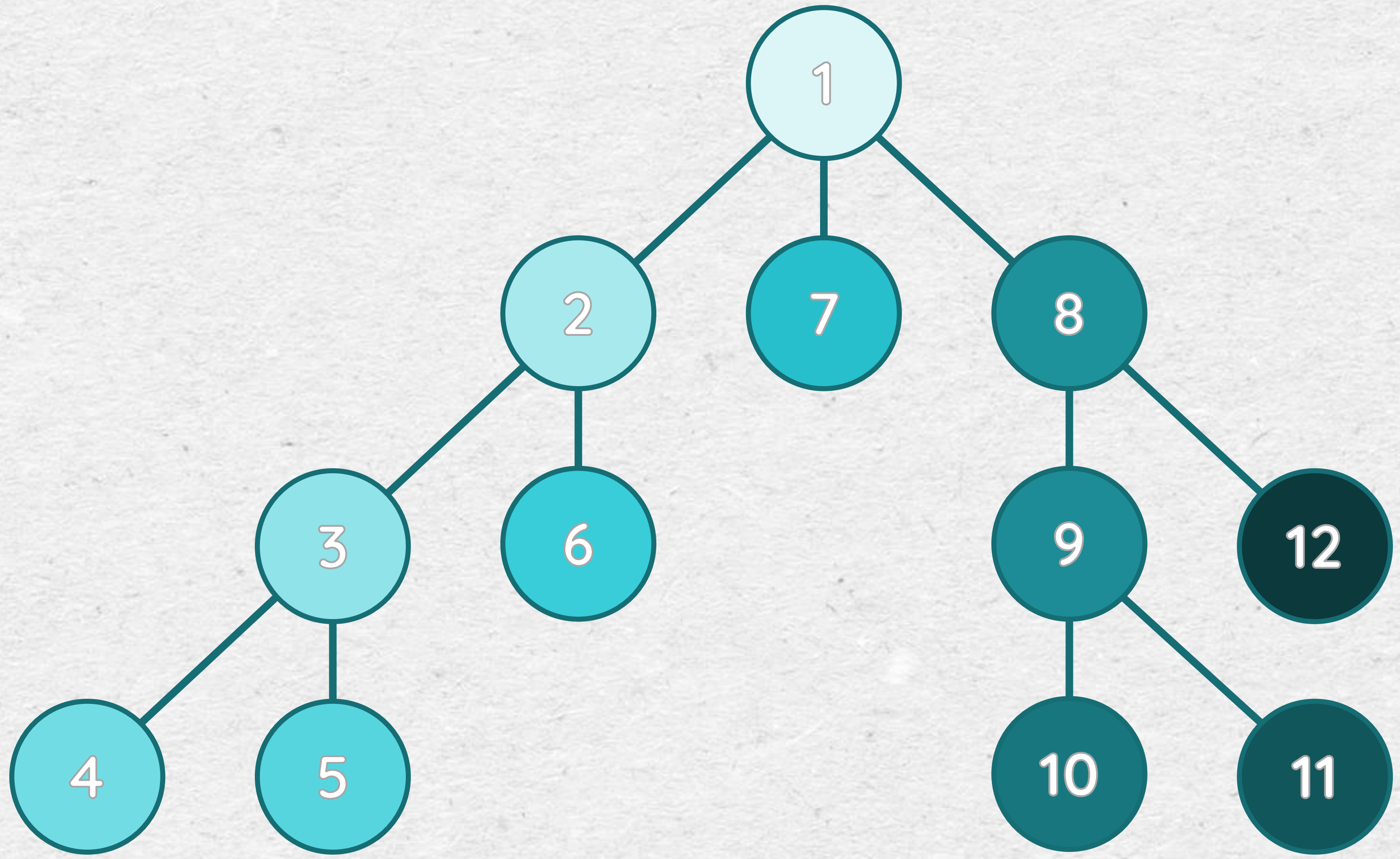
Adjacency-matrix

$\Theta(V^2)$

Traversal Algorithms



Breadth-first search



Depth-first search

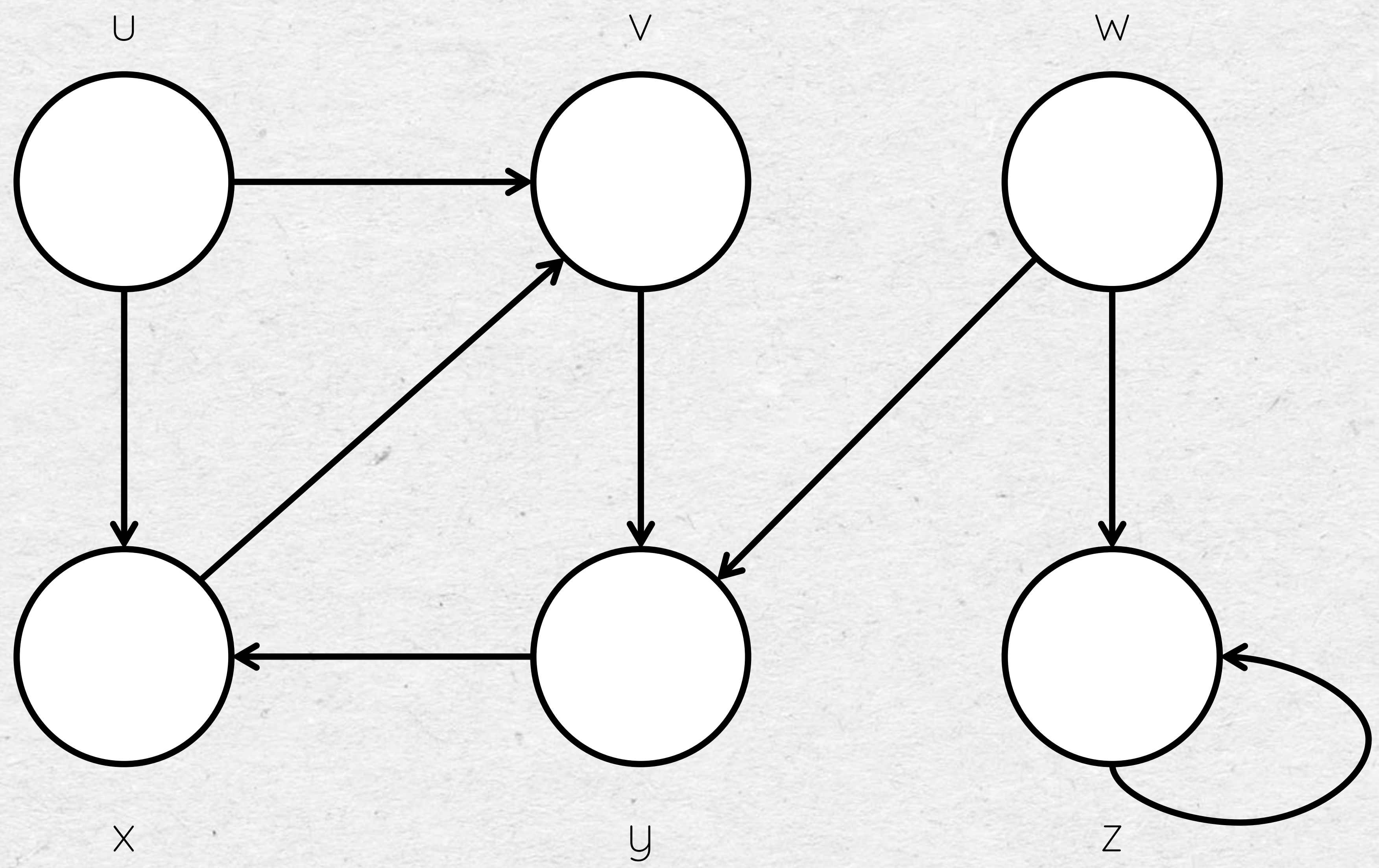
Depth-first Search Algorithm

```
g = Graph
g.properties = [
    cyclic,
    unweighted,
    sparse,
    directed
]
```

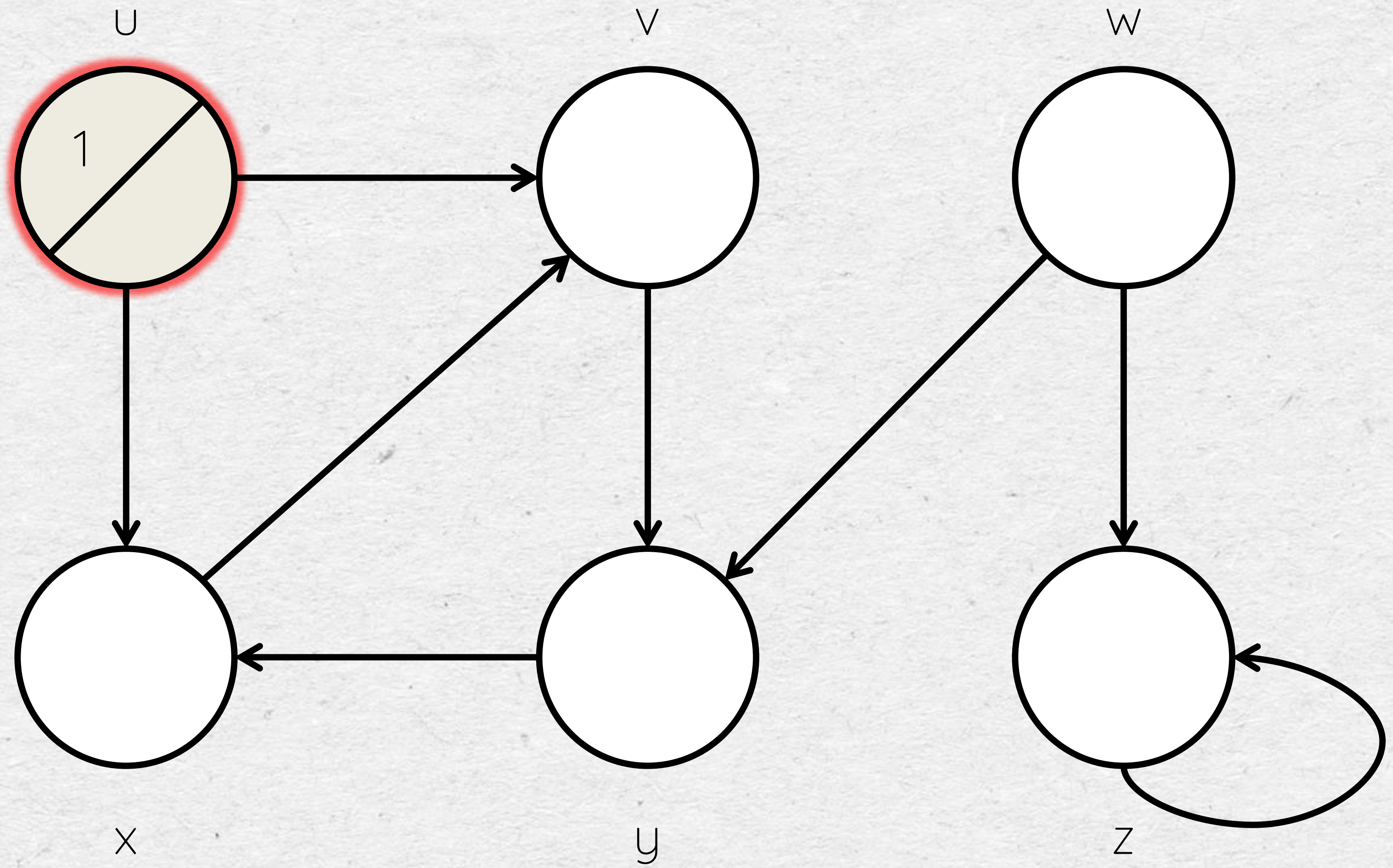
```
v = Vertex
v.color = WHITE
v.Π = NIL
v.d = NIL
v.f = NIL
```

```
e = Edge
e.types =
    tree edges as T or
    back edges as B or
    forward edges as F or
    cross edges as C
```

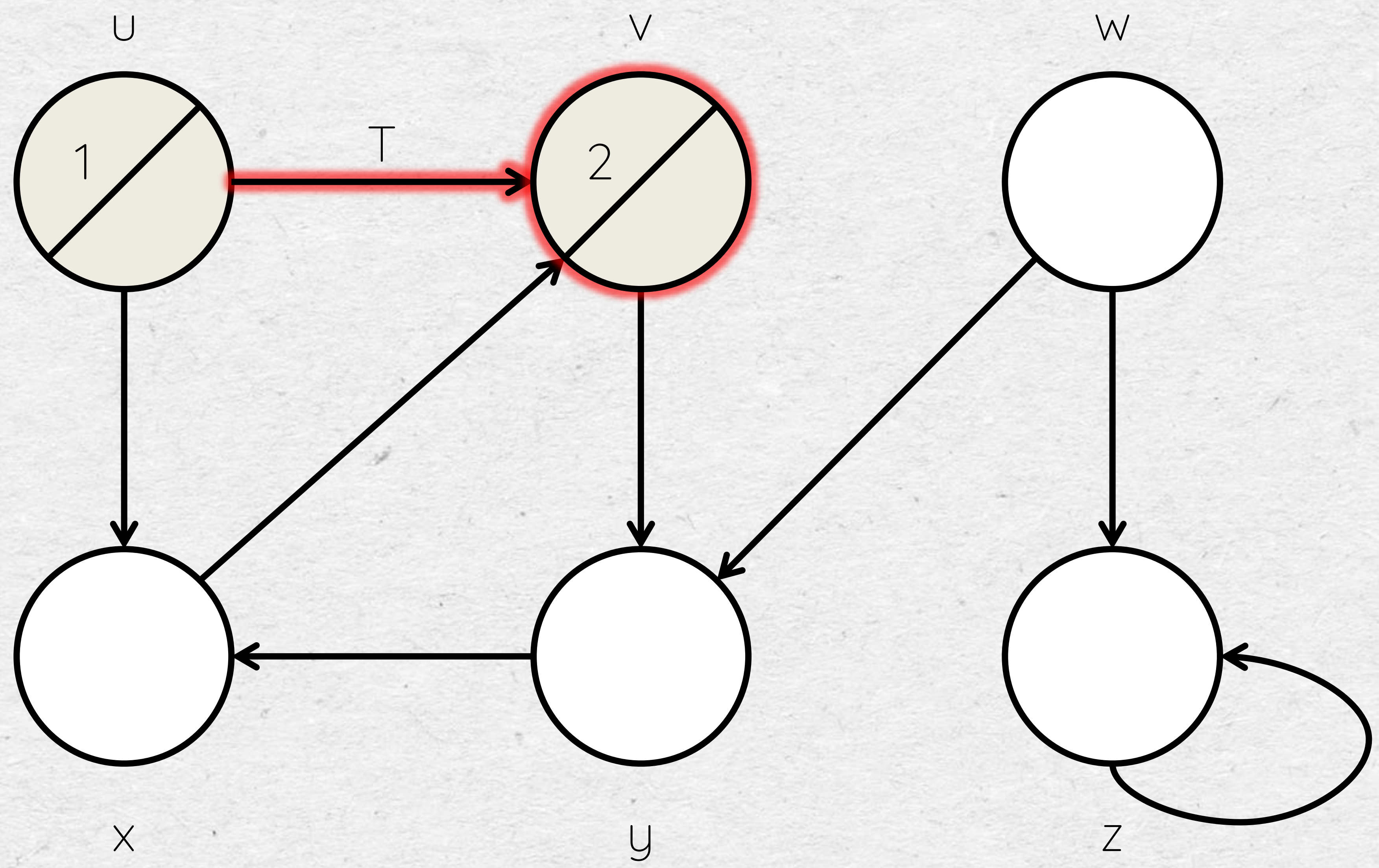
Depth-first Search Algorithm



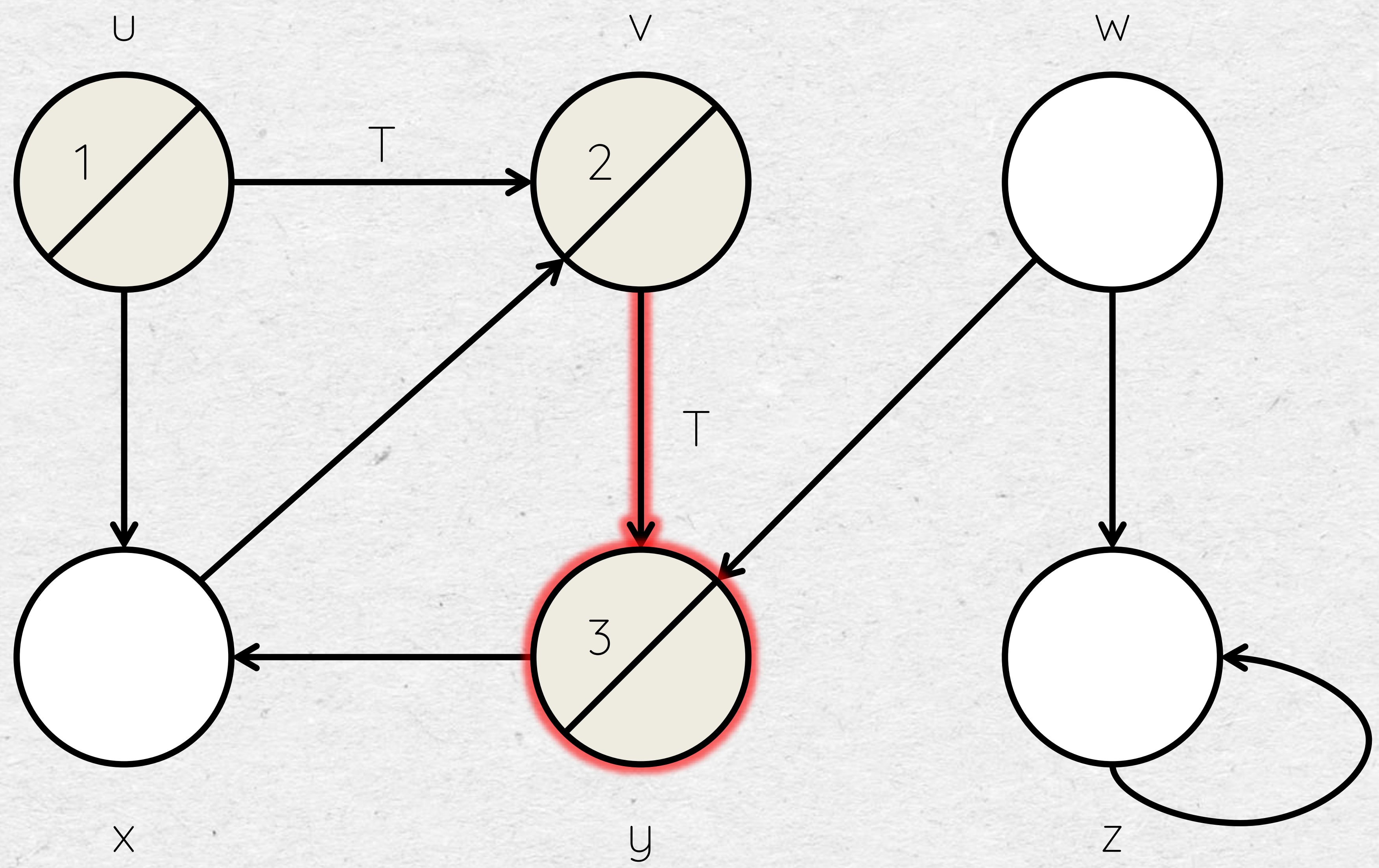
Depth-first Search Algorithm



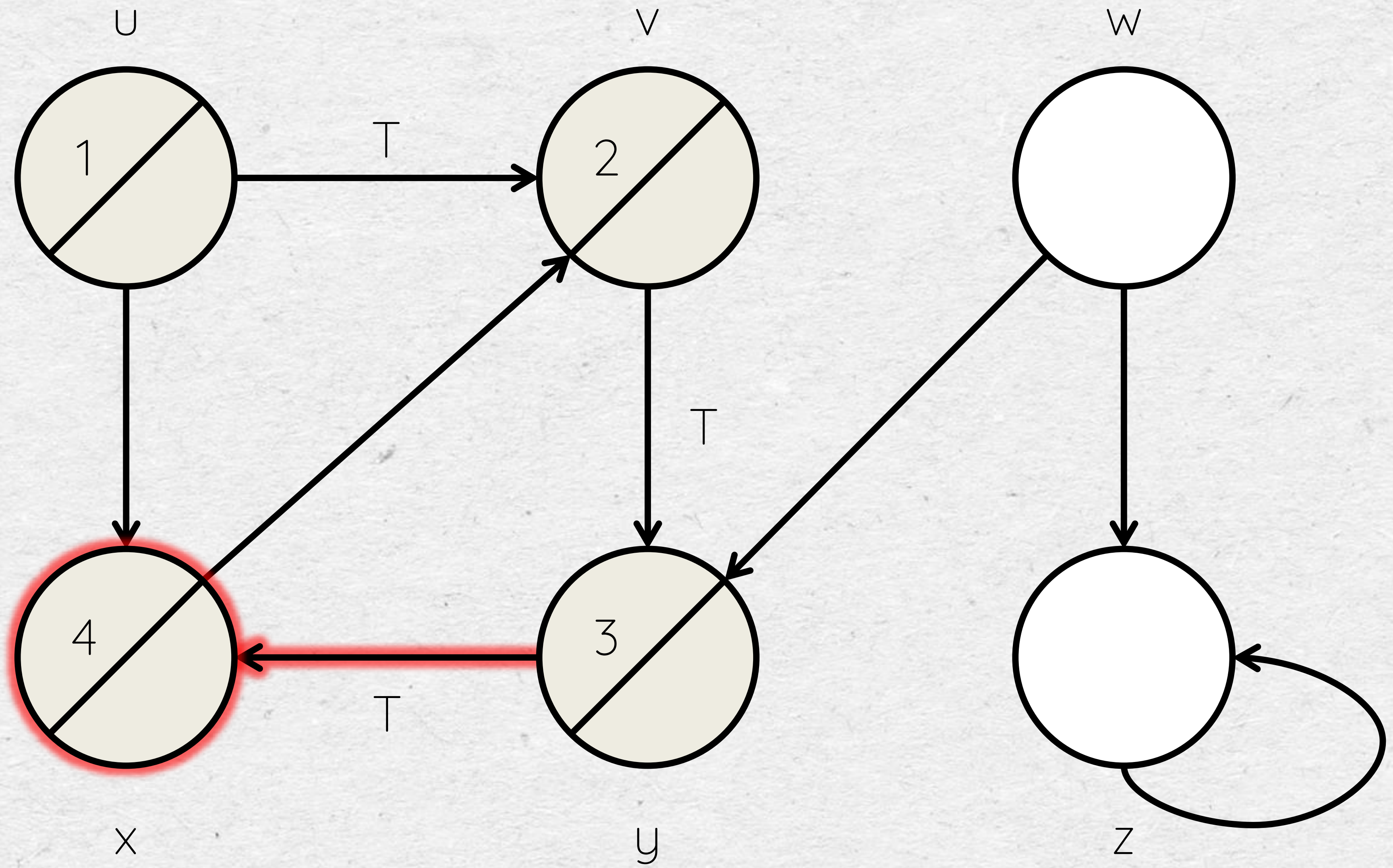
Depth-first Search Algorithm



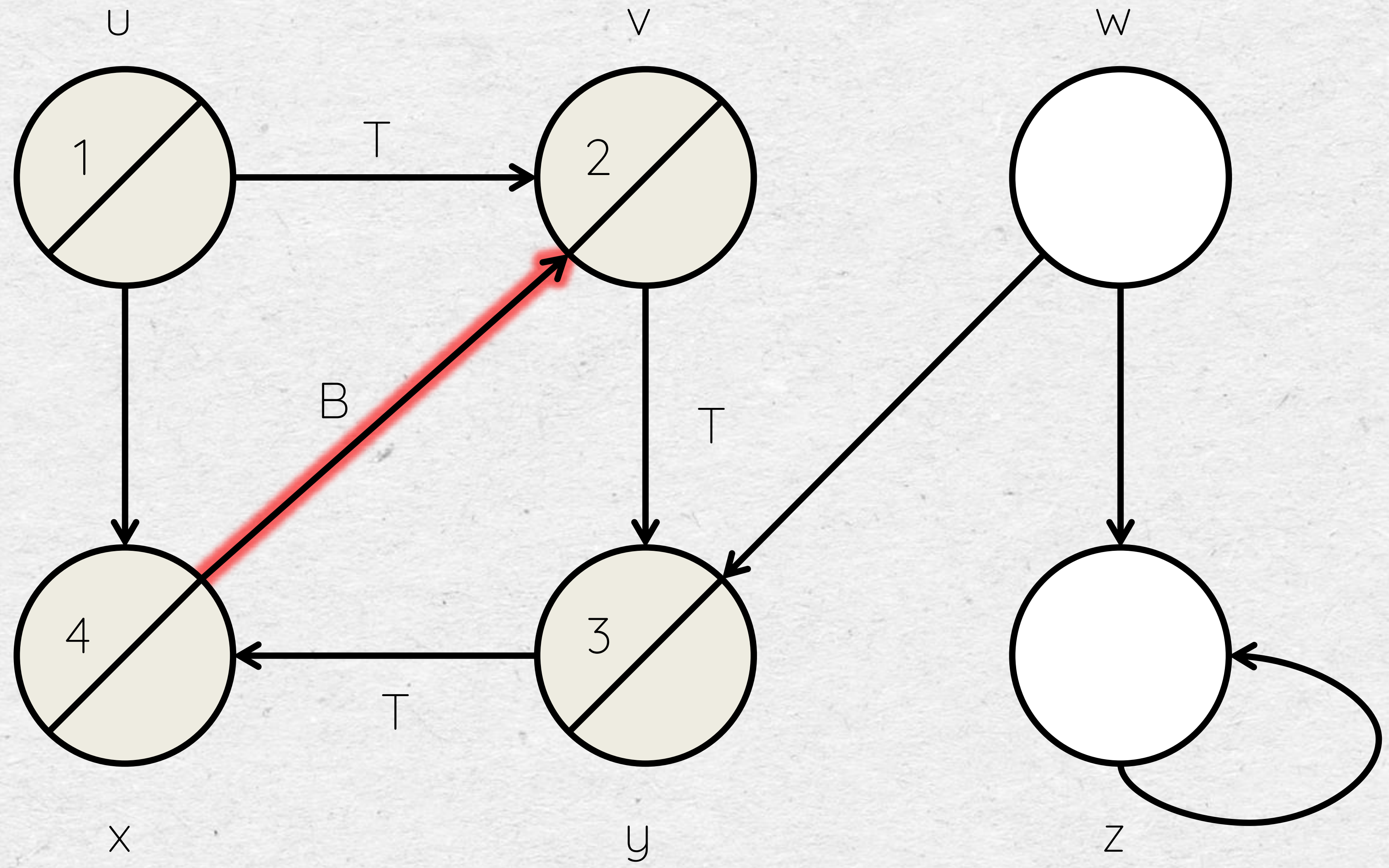
Depth-first Search Algorithm



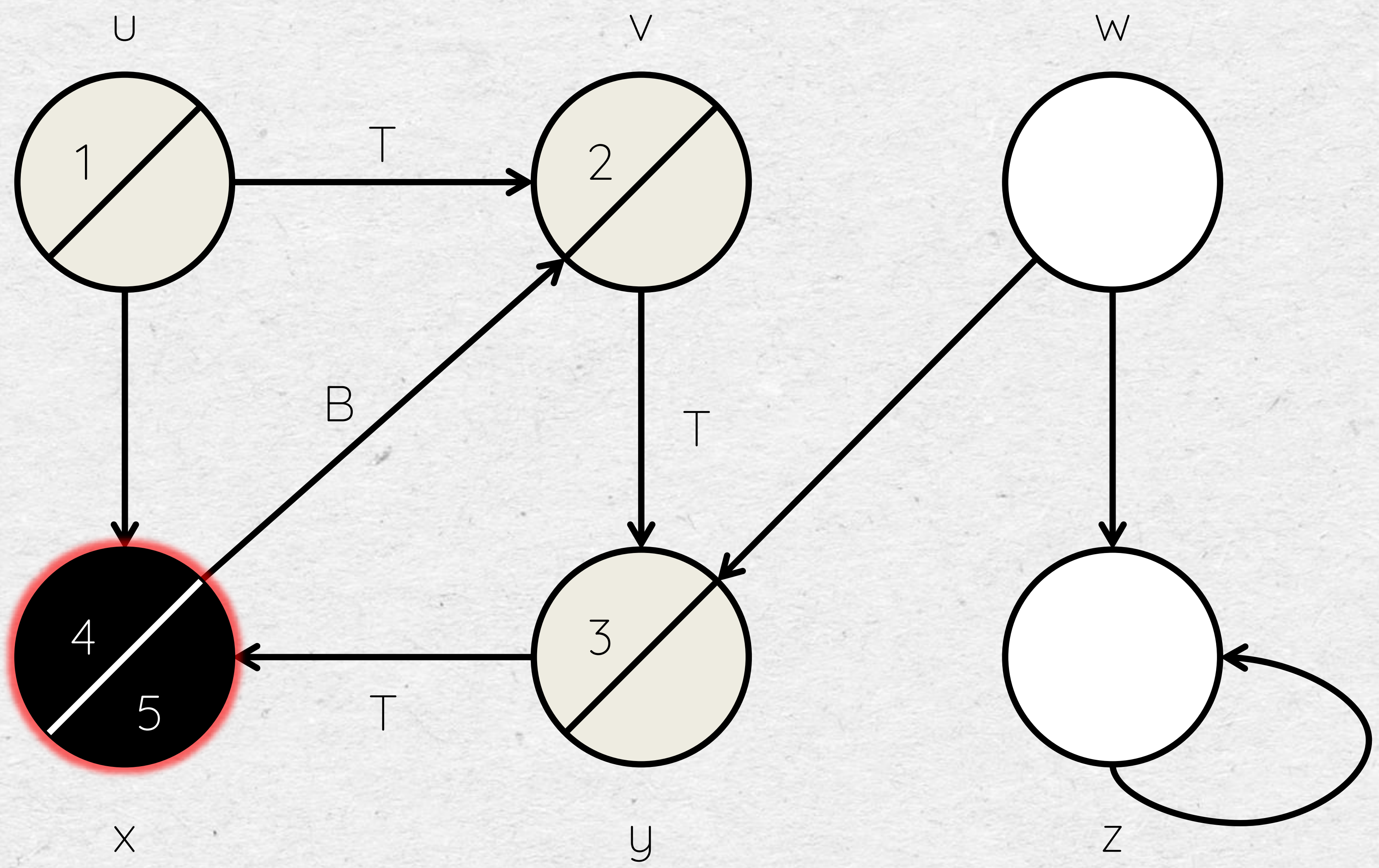
Depth-first Search Algorithm



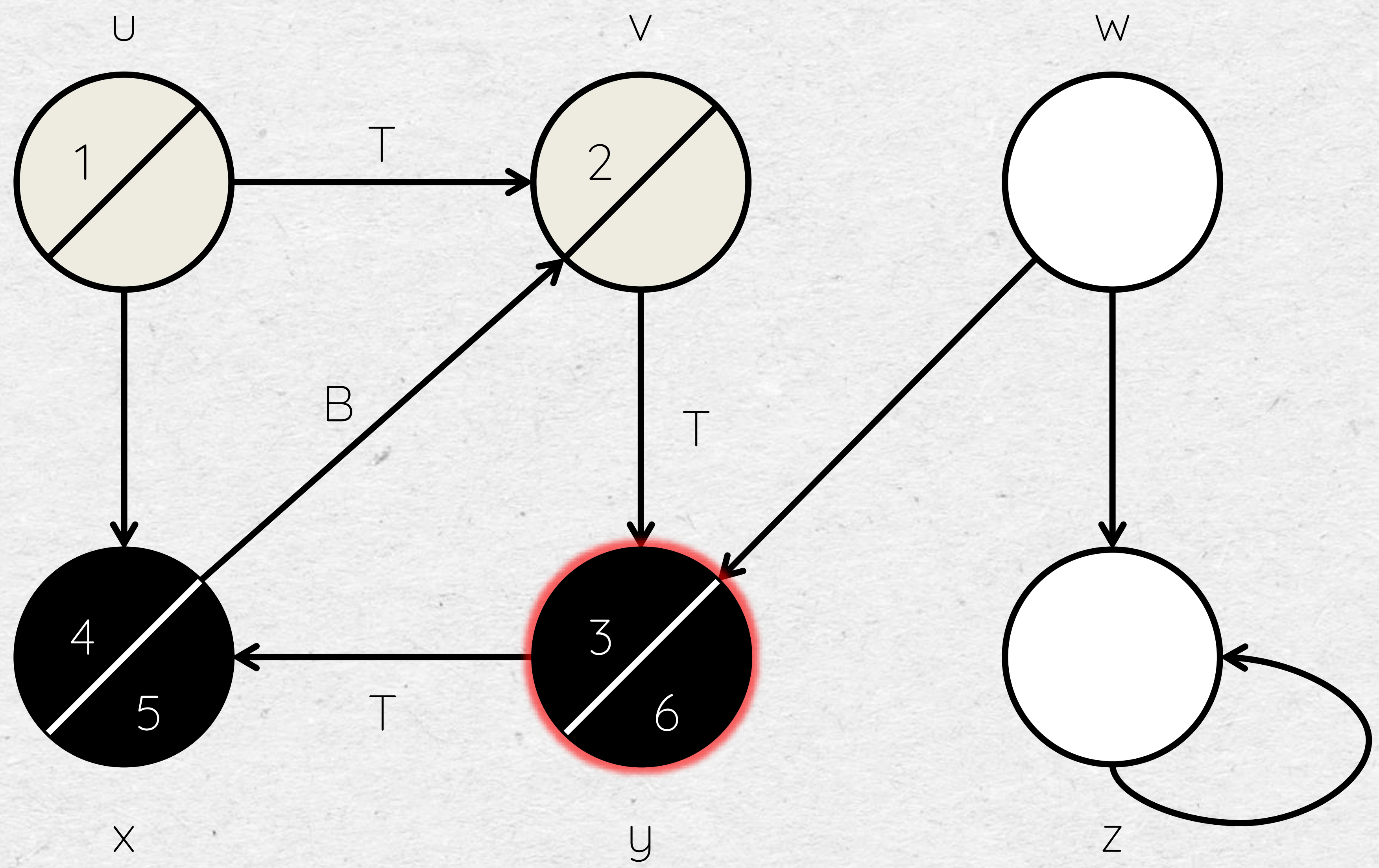
Depth-first Search Algorithm



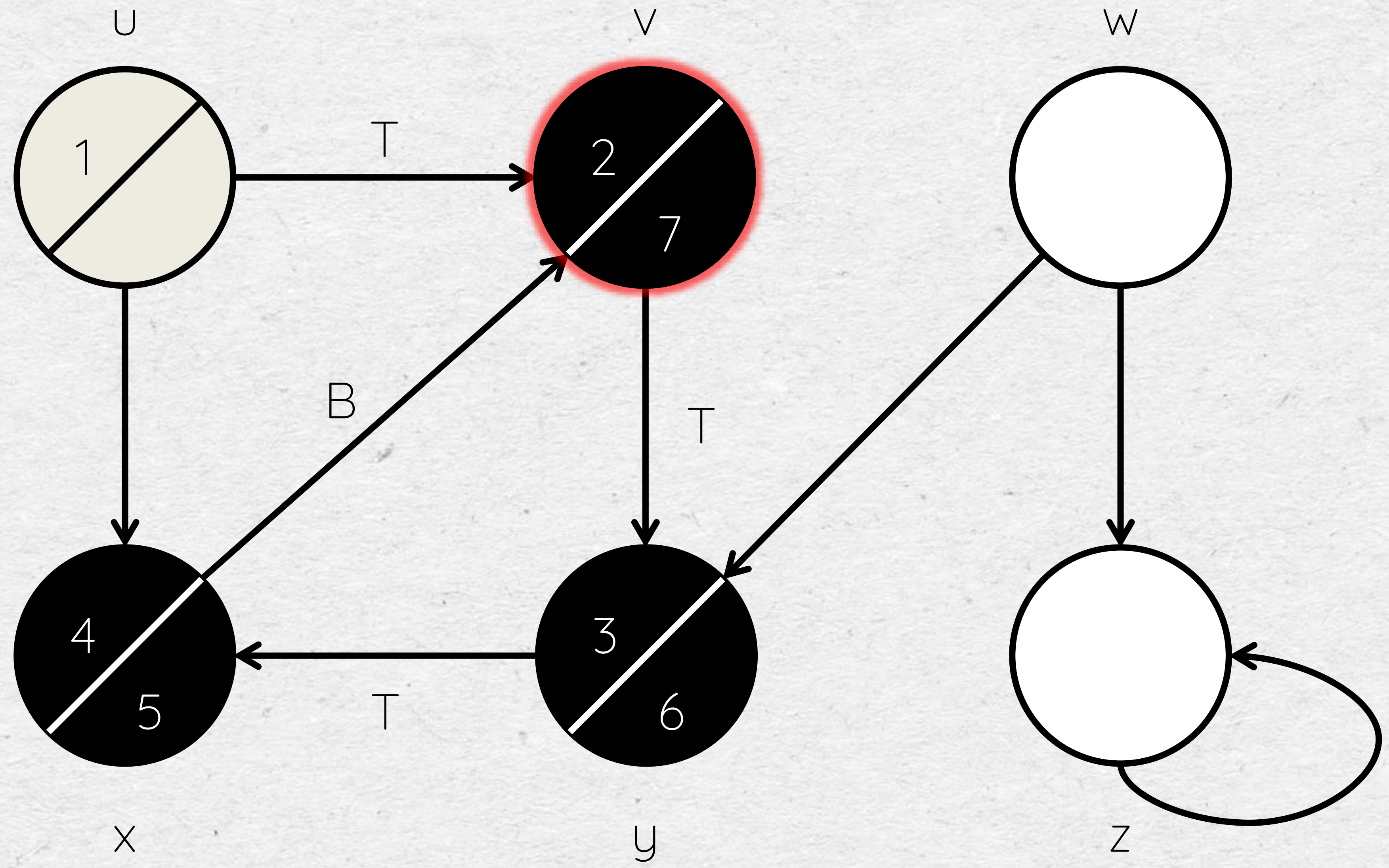
Depth-first Search Algorithm



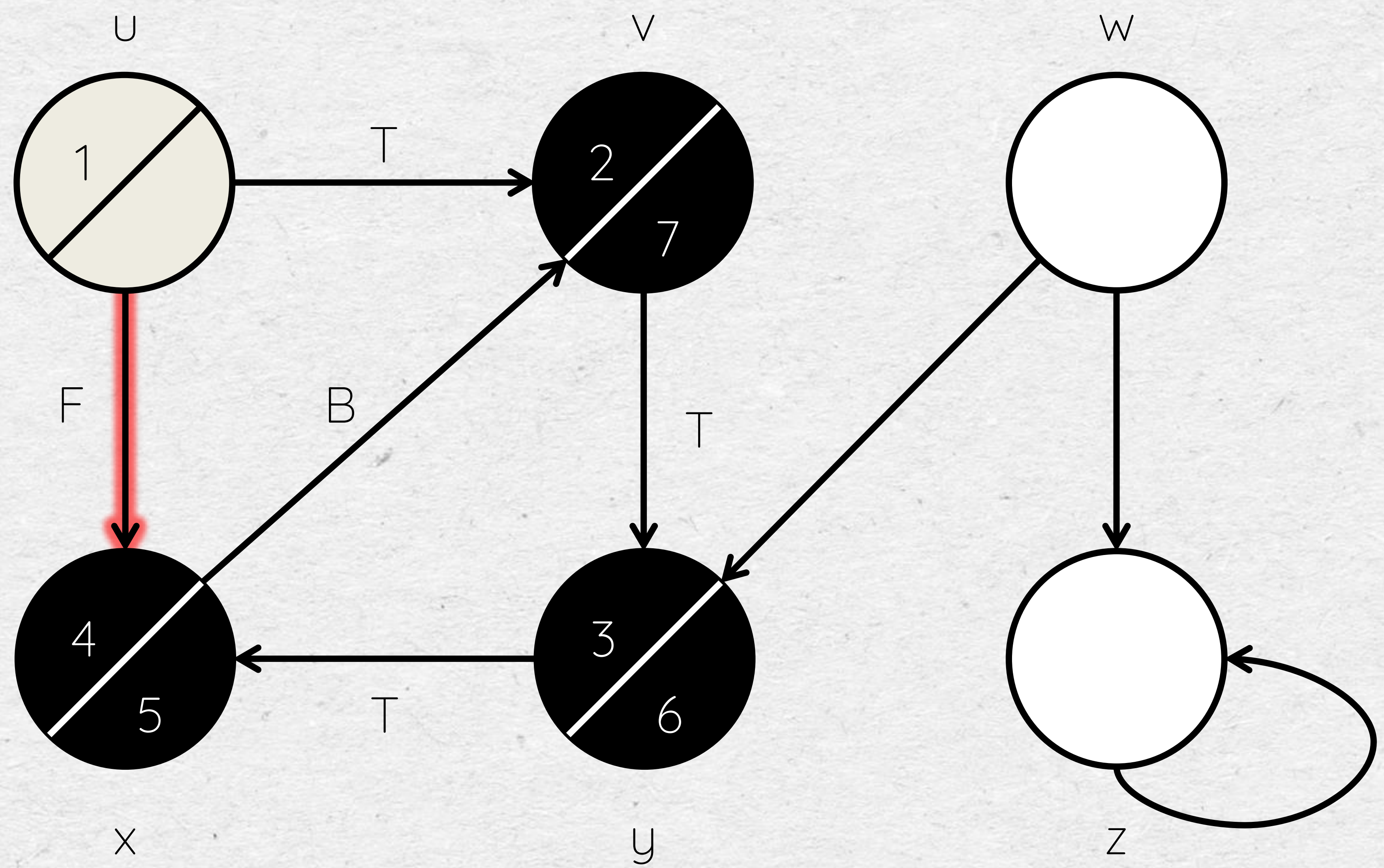
Depth-first Search Algorithm



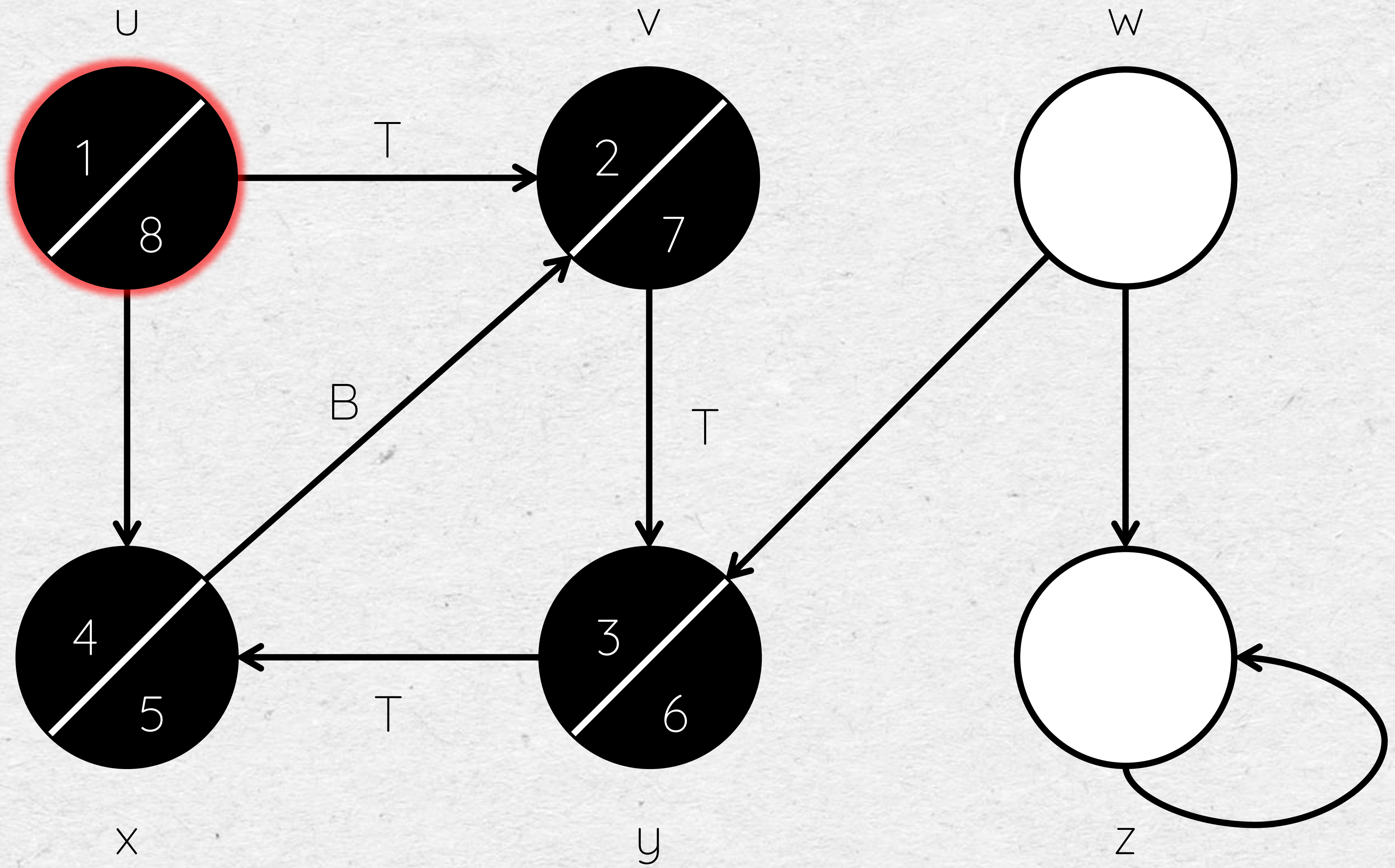
Depth-first Search Algorithm



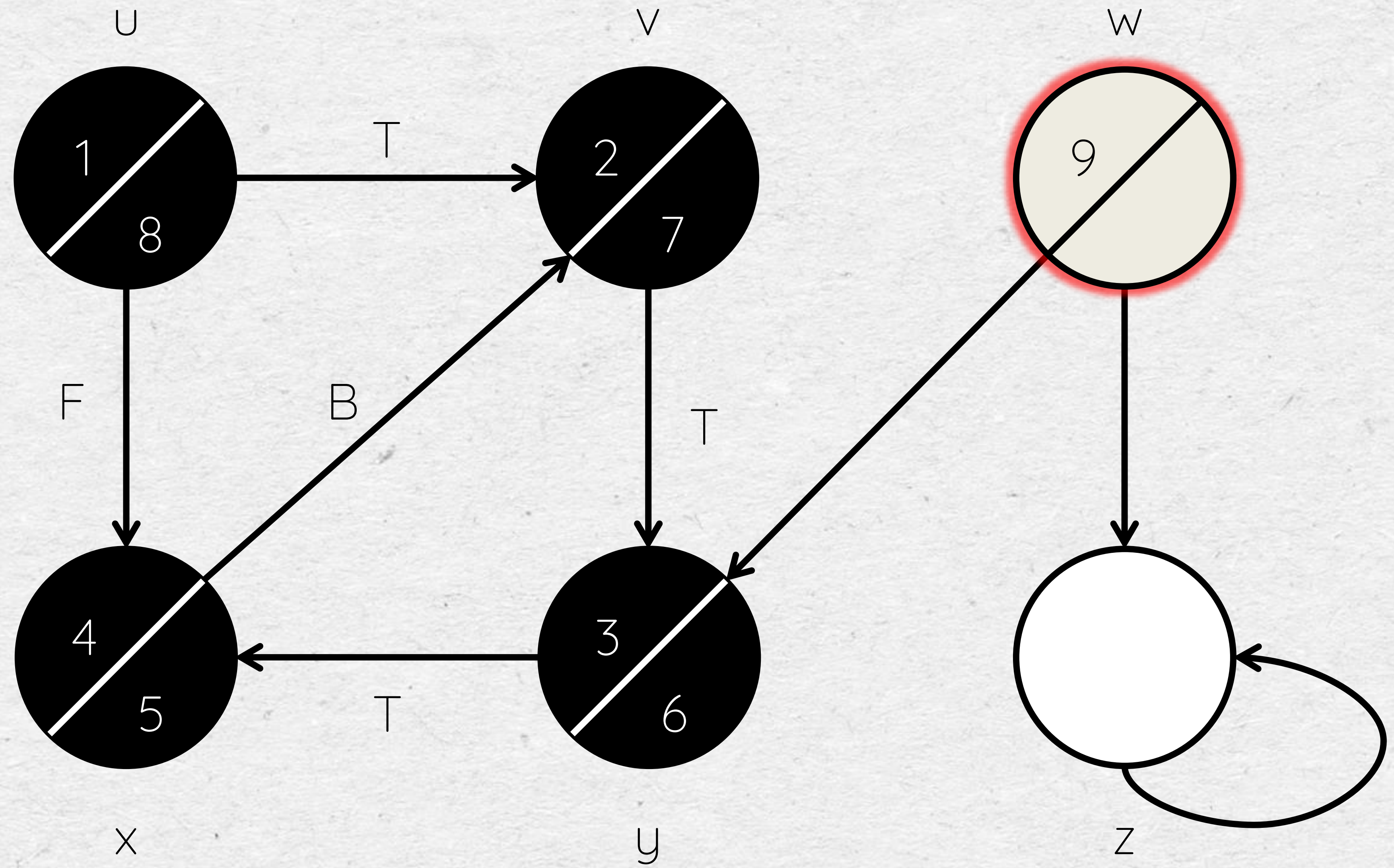
Depth-first Search Algorithm



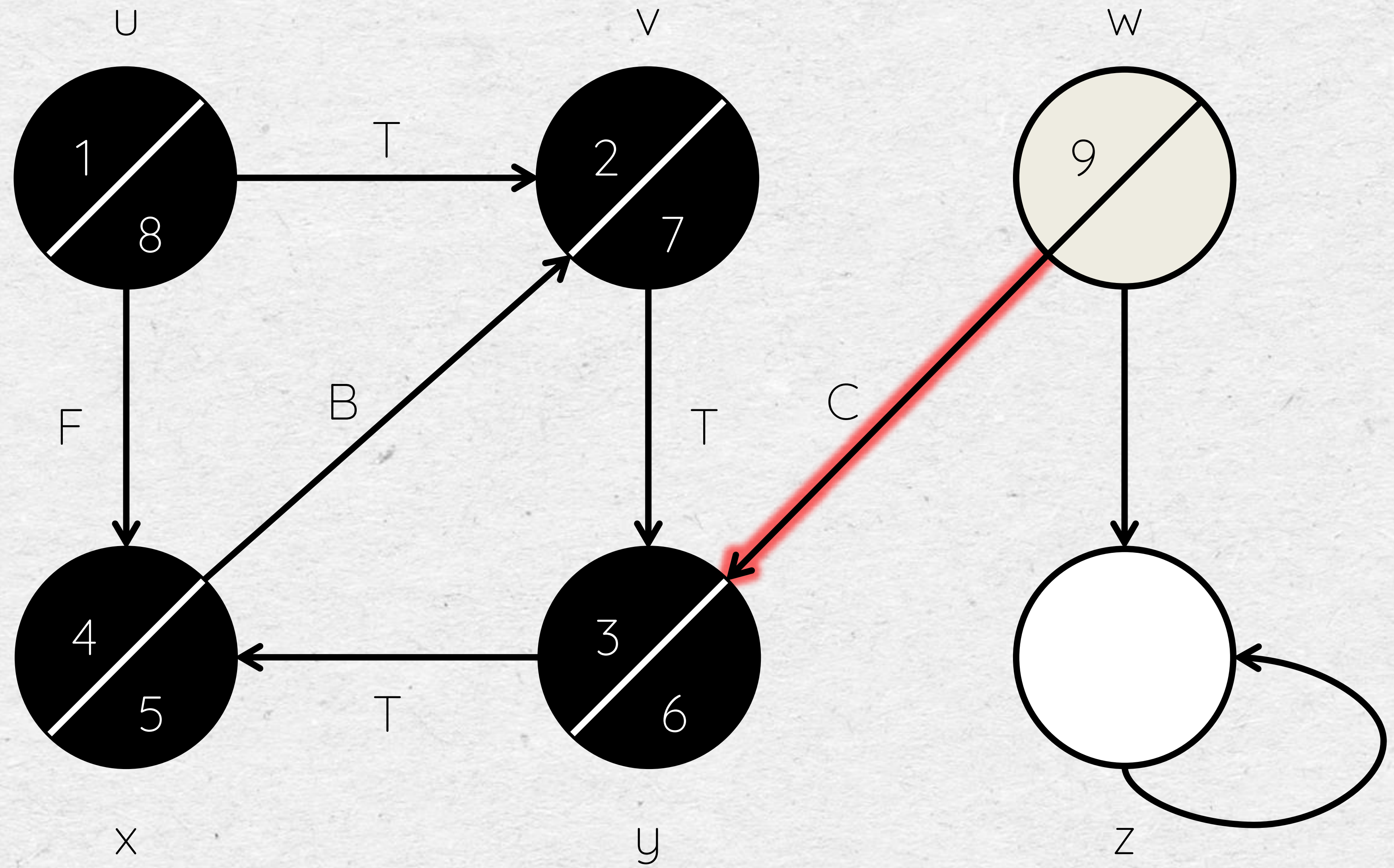
Depth-first Search Algorithm



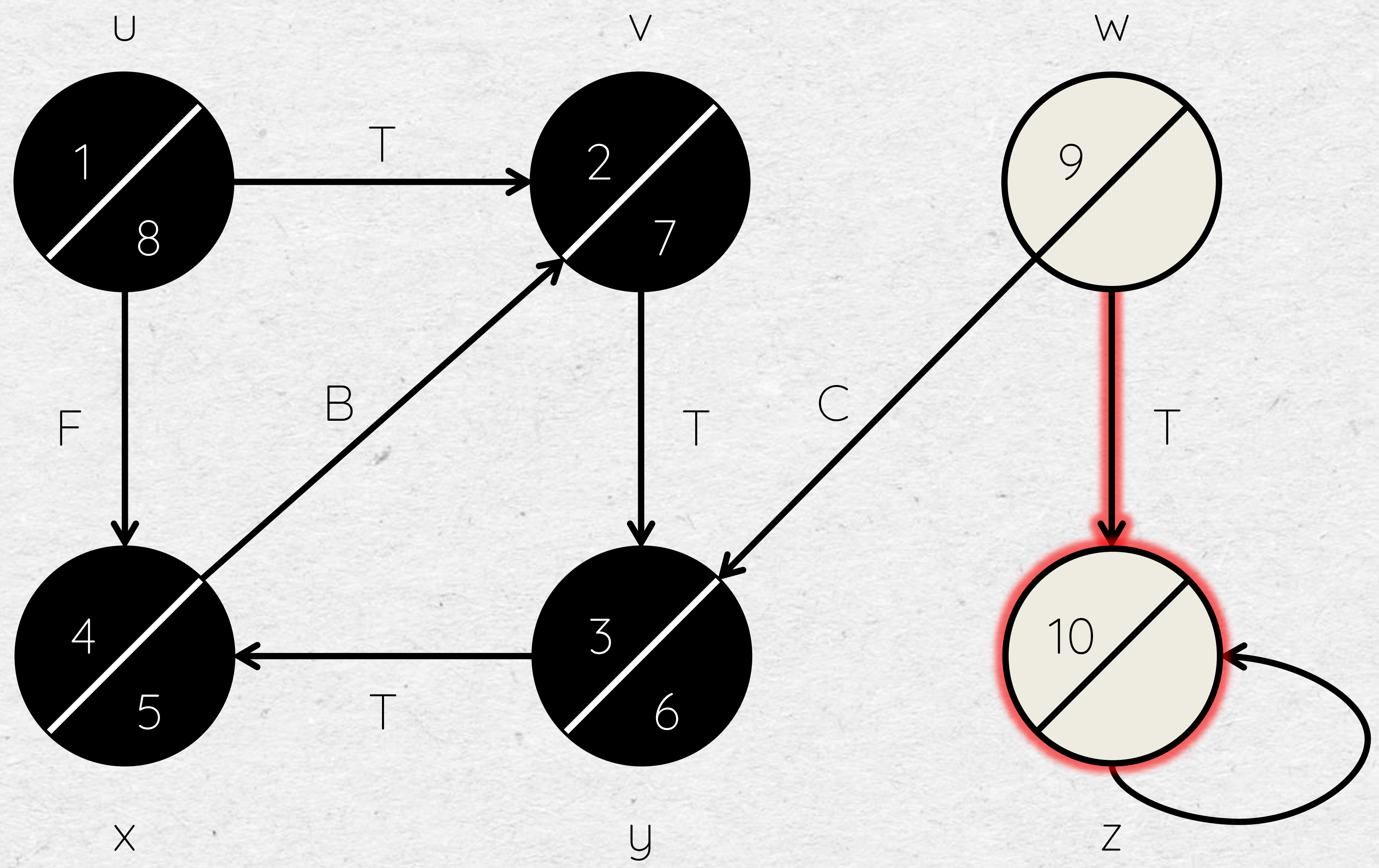
Depth-first Search Algorithm



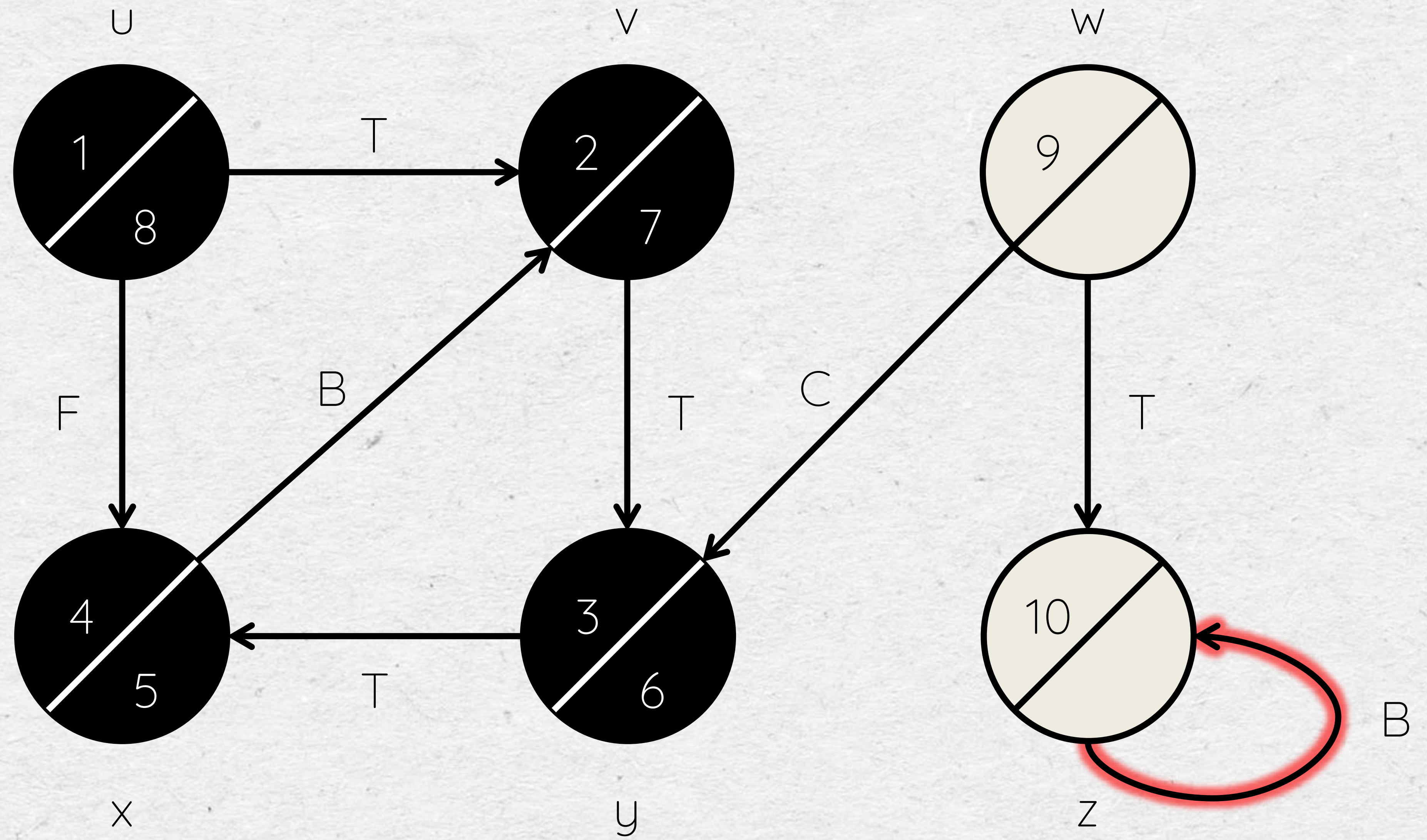
Depth-first Search Algorithm



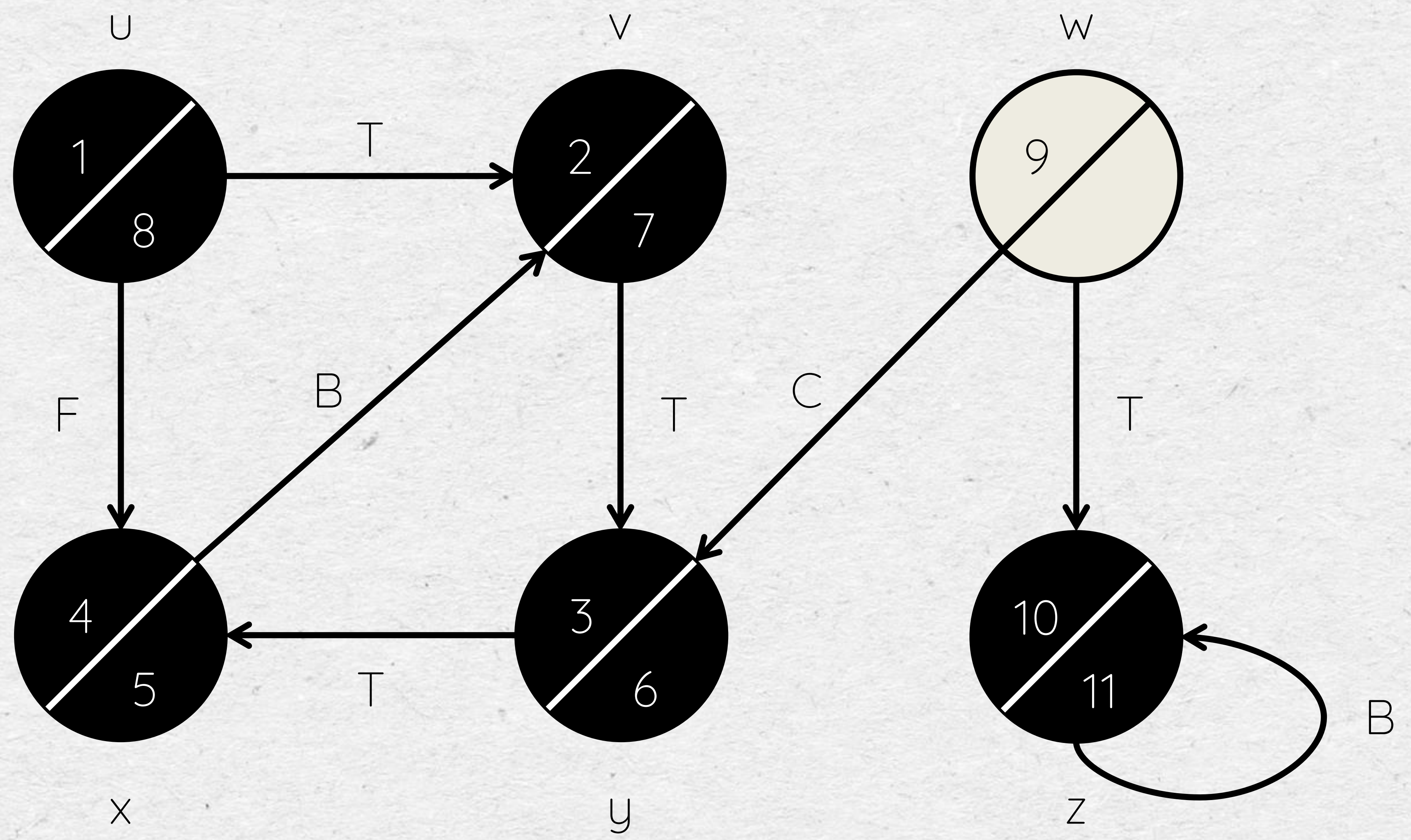
Depth-first Search Algorithm



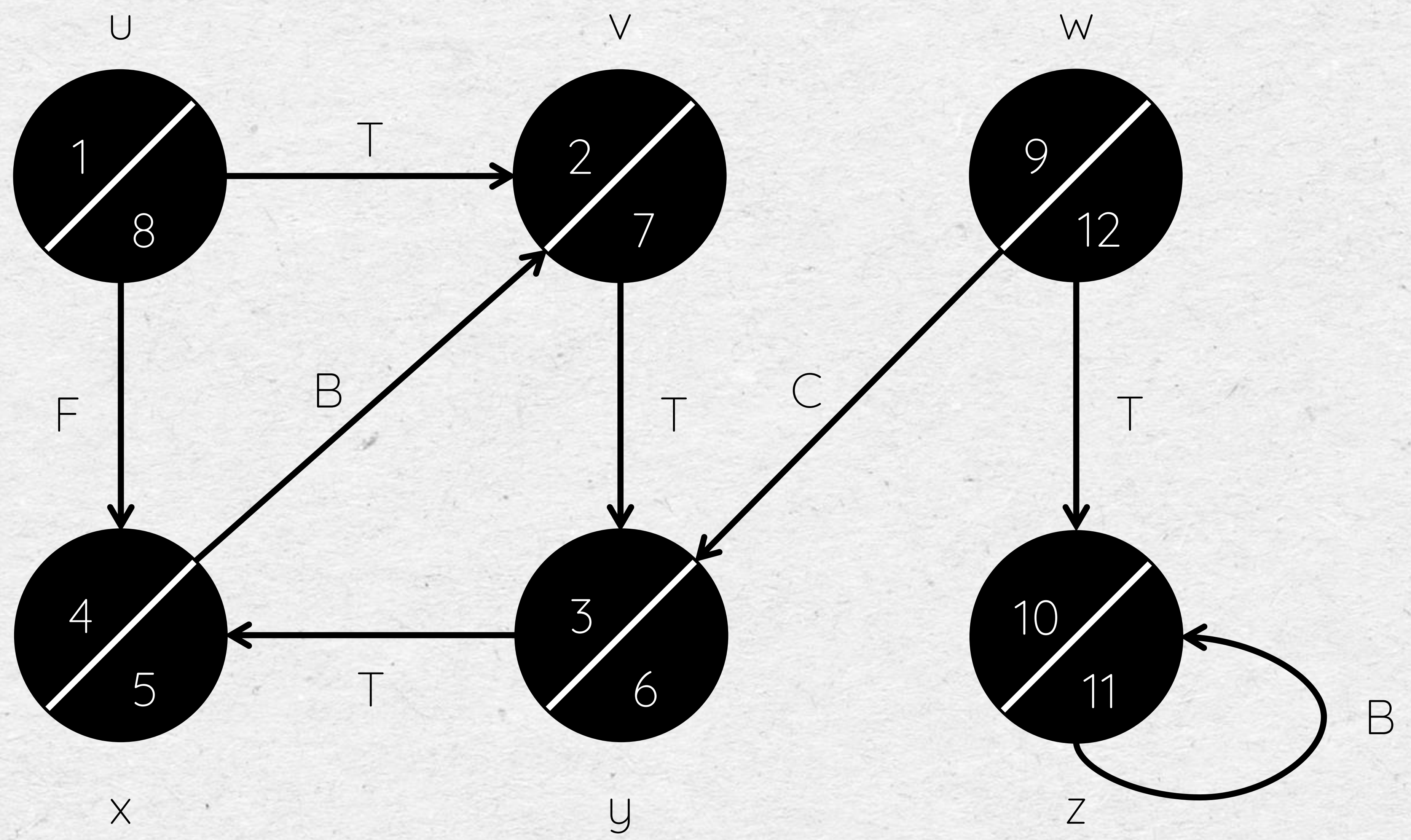
Depth-first Search Algorithm



Depth-first Search Algorithm



Depth-first Search Algorithm

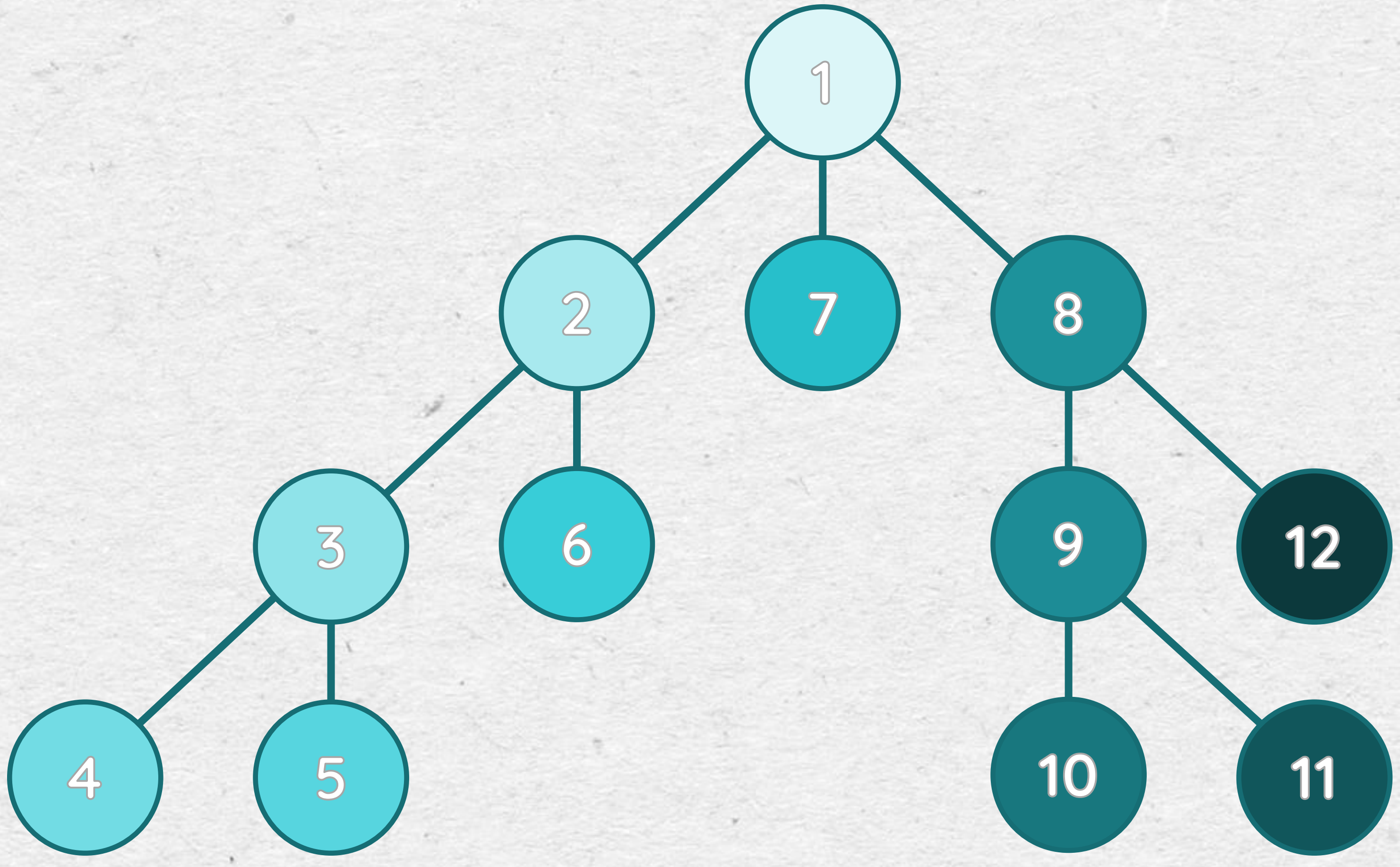


Depth-first Search Algorithm

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

Depth-first Search Algorithm



Depth-first Search Algorithm

```
DFS(G)                                          costtimes
for each vertex $u \in G.V$                     $V + 1$
    $u.color$ = WHITE                           $V$
    $u.\pi$ = NIL                               $V$
$time$ = 0                                      $1$
for each vertex $u \in G.V$                     $V + 1$
    if $u.color$ == WHITE                       $V$
        DFS-VISIT(G, $u$)                       $\sum_{0}^{V} 1$
```

E.length = $\sum_{0}^{G.Adj[V]} 1$

$T(G) = \Theta(V + E)$

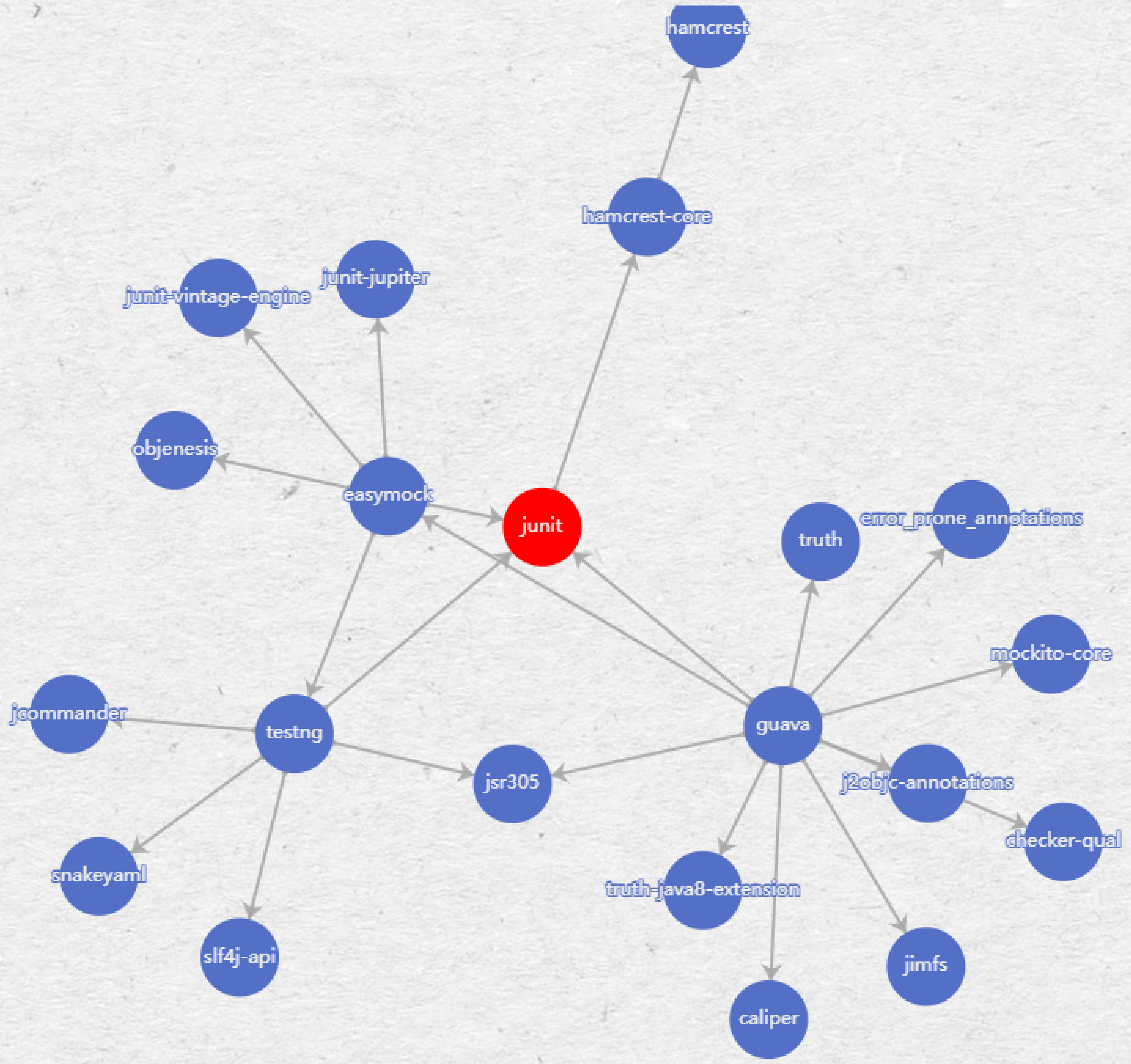
```
DFS-VISIT(G, $u$)                               costtimes
$time$ = $time$ + 1                             $1$
$u.d$ = $time$                                  $1$
$u.color$ = GRAY                                $1$
for each vertex $v$ in G.$Adj$[$u$]            $G.Adj[u] + 1$
    if $v.color$ == WHITE                       $G.Adj[u]$
        $v.\pi$ = u                             $\sum_{0}^{G.Adj[u]} 1$
        DFS-VISIT(G, $v$)                       $\sum_{0}^{G.Adj[u]} 1$
$time$ = $time$ + 1                             $1$
$u.f$ = $time$                                  $1$
$u.color$ = BLACK                               $1$
```

Depth-first Search Algorithm

IMPLEMENTATION

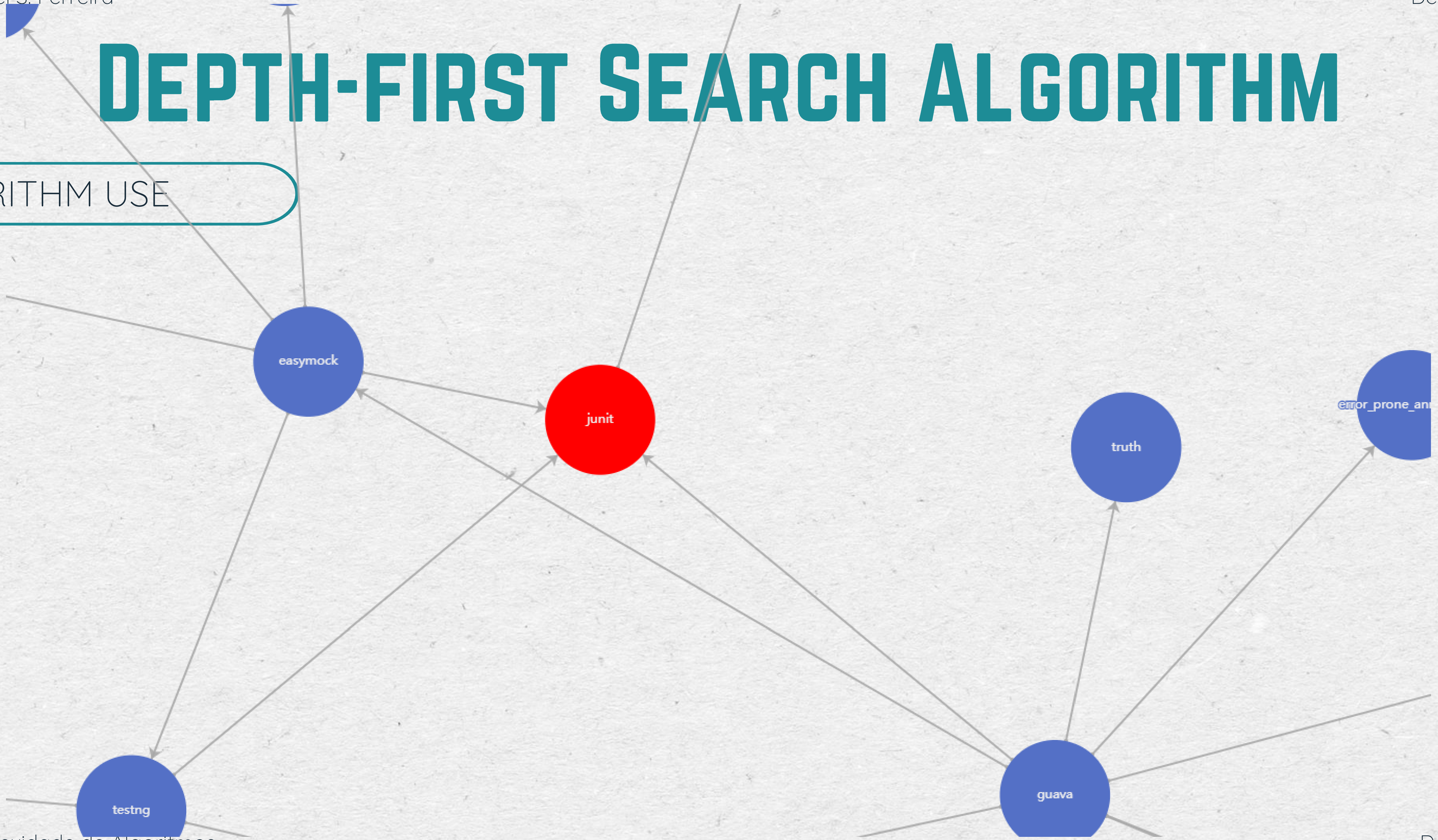
Depth-first Search Algorithm

ALGORITHM USE



Depth-first Search Algorithm

ALGORITHM USE



Depth-first Search Algorithm

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

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

To Remember

01 Properties (Sparce, Cyclic, Directed, Weighted)

02 Representation (Adjacency-List or Adjacency-Matrix)

$\Theta(V + E)$ $\Theta(V^2)$

03 Traversal algorithms (Breath-first search and Depth-first search)

04 Complexity ($T(G) = \Theta(V + E)$)

05 Implementation (Adjusts needed in some cases)

QUESTIONS OR COMMENTS?

Thanks!!