Basic Graph Algorithms

Lecture 06

Ritu Kundu

King's College London

Tue, Nov 07, 2017

1. Introduction
    - Representation
    - Graph Traversal
2. BFS
3. DFS
    - Application: Topological Sort
    - Application: Strongly Connected Components

Outline

1. Introduction
    - Representation
    - Graph Traversal
2. BFS
3. DFS
    - Application: Topological Sort
    - Application: Strongly Connected Components

Graph

A Graph $G = (V, E)$ consists of a set V of vertices(nodes) and a set E of edges(arcs). $(u, v) \in E$, where $u, v \in V$, denotes an edge between vertex u and vertex v. $|V| = n, |E| = m$

Types: Based on number of edges

- Dense
- Sparse

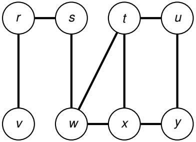
Motivation

- Connection Problems
- Trasportation Problems
- Scheduling Problems
- Network Analysis (Visualisation)

Definitions

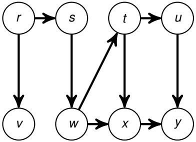
Undirected graph

A graph with undirected edges $[(u, v) = (v, u)]$.



Directed graph (Digraph)

A graph with directed edges $[(u, v) \neq (v, u)]$.



Definitions

Adjacent vertices

Vertices u and v are *adjacent vertices* iff (u, v) is an edge in the graph.

Incident edge

The edge (u, v) is *incident* on vertices u and v.

Degree

Let G be an undirected graph.
The *degree* d_u of a vertex u is the number of edges incident on u.

In-degree/ Out-degree

Let G be a directed graph.
The *in-degree* d_u^{in} of a vertex u is the number of edges incident to u (incoming edges).
The *out-degree* d_u^{out} of a vertex u is the number of edges incident from u (outgoing edges).

Definitions

Path

A *path* P from vertex v_1 to vertex v_k is a sequence of vertices $P =< v_1, v_2, ..., v_k >$ such that $(v_i, v_{i+1}) \in E \forall i = [1..k]$. P is said to be *simple* iff the vertices are unique.

Cycle

A cycle is a path such that $v_1 = v_k$. A cycle is said to be simple if vertices (except first and last) are unique.

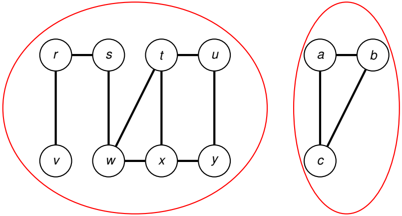
DAG

A Directed Acyclic Graph(DAG) is a directed graph without cycles.

Definitions

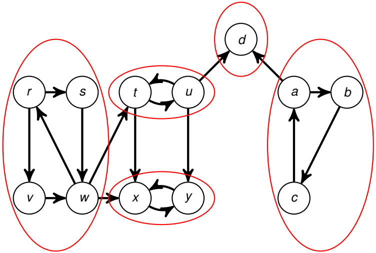
Connected components

A *connected component* of an undirected graph is a **maximal** subset of the vertices such that for every pair *u* and *v* of vertices in this subset, there is a path from *u* to *v*.



Strongly Connected components

A *strongly connected component* of a directed graph is a **maximal** subset of the vertices such that for every pair *u* and *v* of vertices in this subset, there is a path from *u* to *v*.



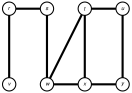
Adjacency Matrix

$n \times n$ matrix A such that

$$A(i,j) = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

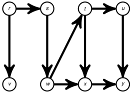
Space Complexity:
Check $(u,v) \in E$:
Traverse neighbours of a vertex u:
Check if G has isolated vertex:
Check if G is complete :
Check if G has a loop :

Example 1: Undirected Graph



	r	s	t	u	v	w	x	y
r	0	1	0	0	1	0	0	0
s	1	0	0	0	0	1	0	0
t	0	0	0	1	0	1	1	0
u	0	0	1	0	0	0	0	1
v	1	0	0	0	0	0	0	0
w	0	1	1	0	0	0	1	0
x	0	0	1	0	0	1	0	1
y	0	0	0	1	0	0	1	0

Example 2: Directed Graph



	r	s	t	u	v	w	x	y
r	0	1	0	0	1	0	0	0
s	0	0	0	0	0	1	0	0
t	0	0	0	1	0	0	1	0
u	0	0	0	0	0	0	0	1
v	0	0	0	0	0	0	0	0
w	0	0	1	0	0	0	1	0
x	0	0	0	0	0	0	0	1
y	0	0	0	0	0	0	0	0

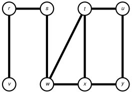
Adjacency Matrix

$n \times n$ matrix A such that

$$A(i,j) = \begin{cases} 1 \text{ if } (i,j) \in E \\ 0 \text{ otherwise} \end{cases}$$

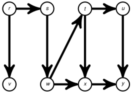
Space Complexity: $O(n^2)$
Check $(u,v) \in E$*:* $O(1)$
Traverse neighbours of a vertex u*:* $O(n)$
Check if G *has isolated vertex:* $O(n^2)$
Check if G *is complete :* $O(n^2)$
Check if G *has a loop :* $O(n)$

Example 1: Undirected Graph



	r	s	t	u	v	w	x	y
r	0	1	0	0	1	0	0	0
s	1	0	0	0	0	1	0	0
t	0	0	0	1	0	1	1	0
u	0	0	1	0	0	0	0	1
v	1	0	0	0	0	0	0	0
w	0	1	1	0	0	0	1	0
x	0	0	1	0	0	1	0	1
y	0	0	0	1	0	0	1	0

Example 2: Directed Graph



	r	s	t	u	v	w	x	y
r	0	1	0	0	1	0	0	0
s	0	0	0	0	0	1	0	0
t	0	0	0	1	0	0	1	0
u	0	0	0	0	0	0	0	1
v	0	0	0	0	0	0	0	0
w	0	0	1	0	0	0	1	0
x	0	0	0	0	0	0	0	1
y	0	0	0	0	0	0	0	0

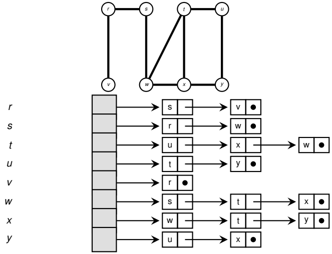
Adjacency List

Array *Adj* of length n such that
$Adj(u)$ is list of vertcis adjacent to u.

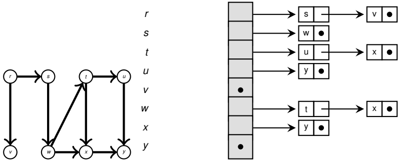
Space Complexity:
Check $(u, v) \in E$*:*
Traverse neighbours of a vertex u*:*

Check if G has isolated vertex:
Check if G is complete :
Check if G has a loop :

Example 1: Undirected Graph



Example 2: Directed Graph

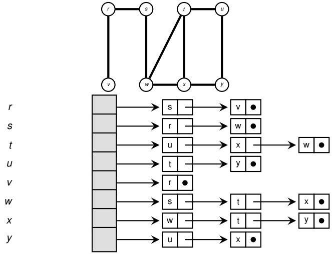


Adjacency List

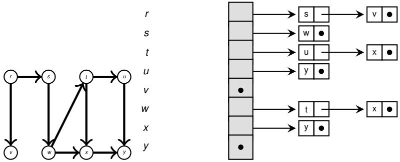
Array *Adj* of length n such that
$Adj(u)$ is list of vertcis adjacent to u.

Space Complexity: $O(n+m)$
Check $(u,v) \in E$: $O(degree_u)$
Traverse neighbours of a vertex u: $O(degree_u)$
Check if G has isolated vertex: $O(n)$
Check if G is complete : $O(n+m)$
Check if G has a loop : $O(n+m)$[What if the list is sorted?]

Example 1: Undirected Graph



Example 2: Directed Graph



Graph Traversal

Problem

Visiting every node of the graph, keeping ‘redundancy’ minimum.

Methods

- Breadth First Search (BFS)
- Depth First Search (DFS)

Color Code

- White: Unvisted
- Gray: Visited at least once (discovered)
- Black: Dead (finished)

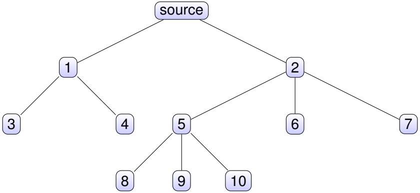
Outline

1. Introduction
   - Representation
   - Graph Traversal
2. BFS
3. DFS
   - Application: Topological Sort
   - Application: Strongly Connected Components

Breadth First Search(BFS)

Visiting Order

Levels: Immediate neighbours, those at 2-hops distance, those at 3-hops distance and so on.



Data Structure used

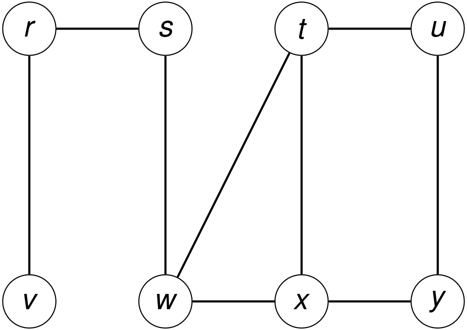
Queue(FIFO)

Algorithm

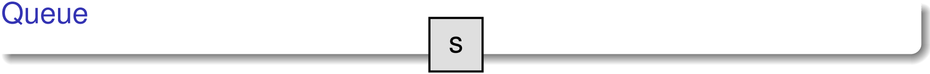
Algorithm 1 BFS Algo

```
procedure BFS($G, s$)   *** $G = (V, E)$ and $s \in V$ ***
    Initialize all nodes as White(unvisited)
    Initialize $s$ as Gray(visited)
    Add $s$ to Queue
    while Queue is not empty do
        Let $u$ be the head of Queue
        for each $v$ adjacent to $u$ do
            if $v$ is unvisted(White) then
                Mark $v$ as Gray(visited)
                Add $v$ to Queue
            end if
        end for
        Remove $u$(head) from Queue
        Mark $u$ as Black(dead)
    end while
end procedure
```

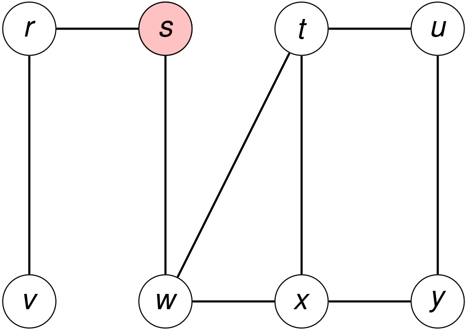
Example

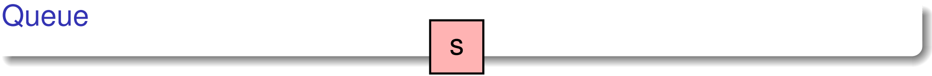


Queue

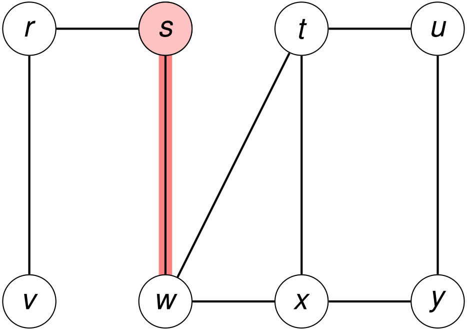


Example



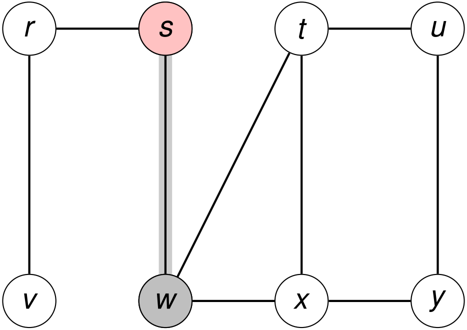


Example

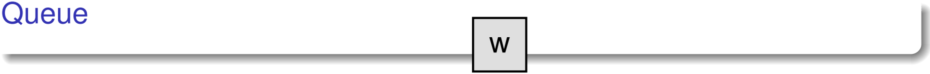


Queue

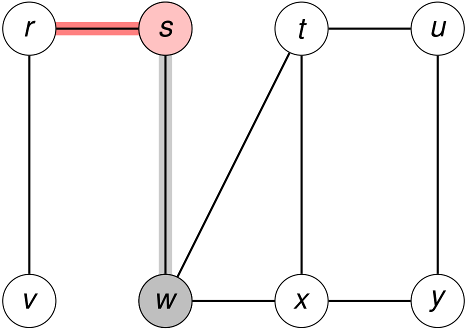
Example



Queue

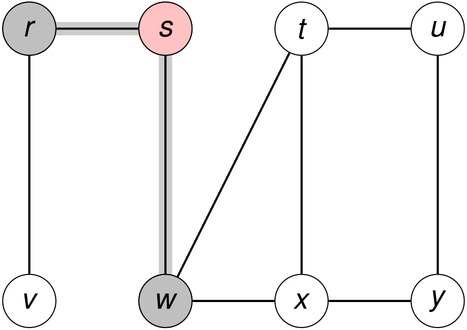


Example



Queue

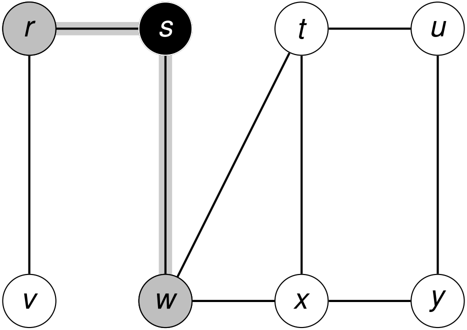
Example



Queue

w	r

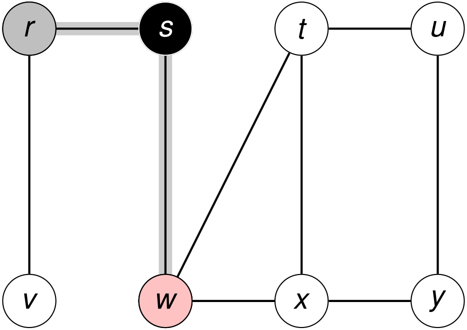
Example



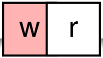
Queue

w	r

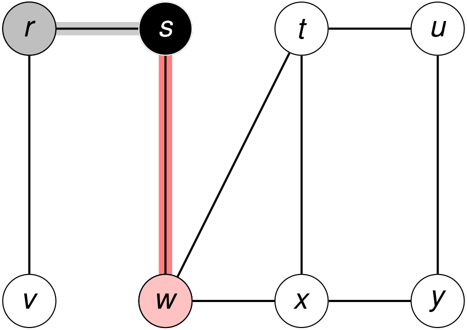
Example



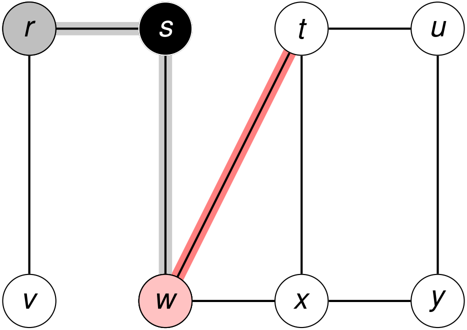
Queue



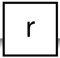
Example



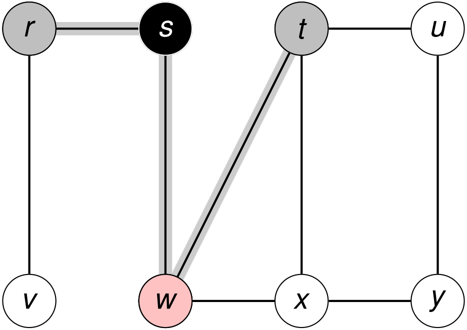
Example



Queue



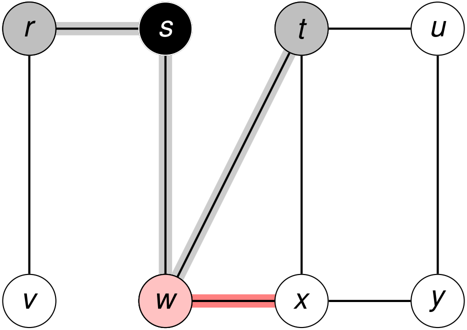
Example



Queue

r	t

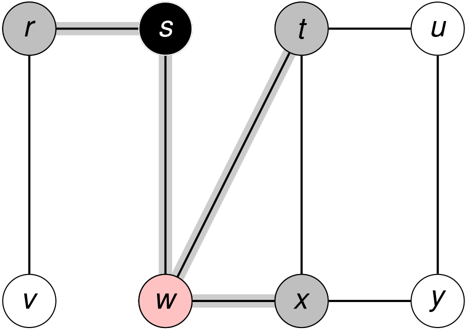
Example



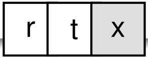
Queue

r	t

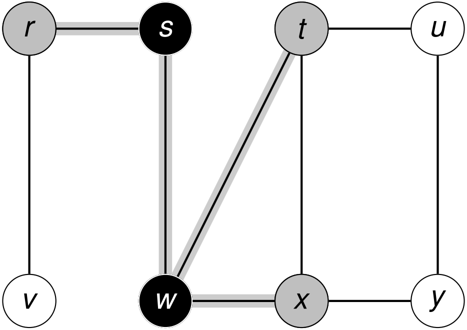
Example



Queue



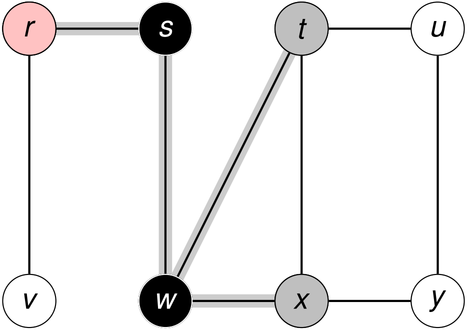
Example



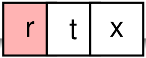
Queue

r	t	x

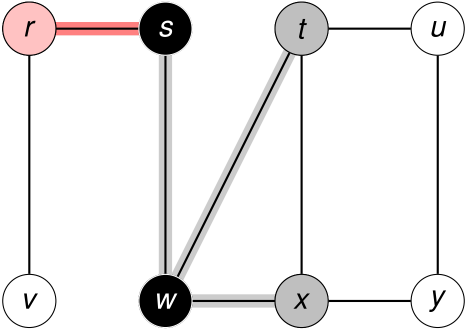
Example



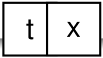
Queue



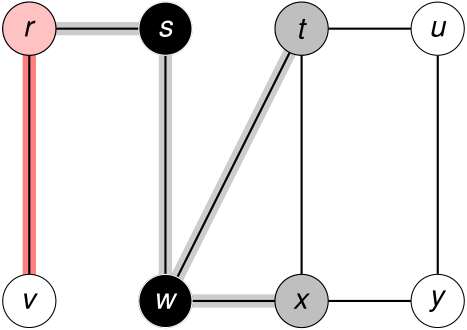
Example



Queue



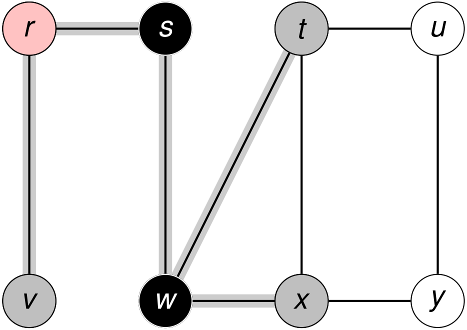
Example



Queue

t	x

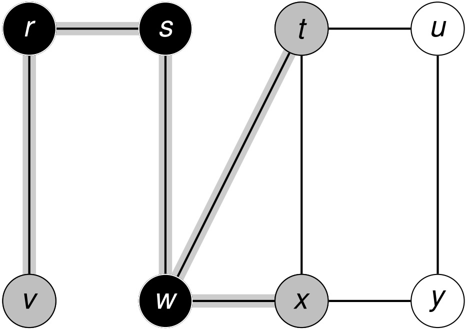
Example



Queue

t	x	v

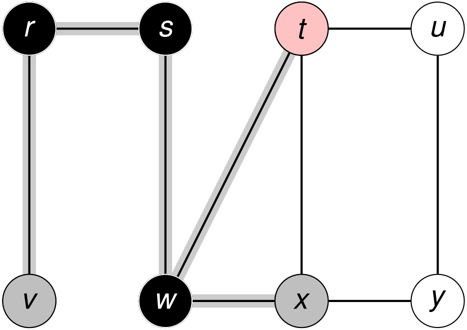
Example



Queue

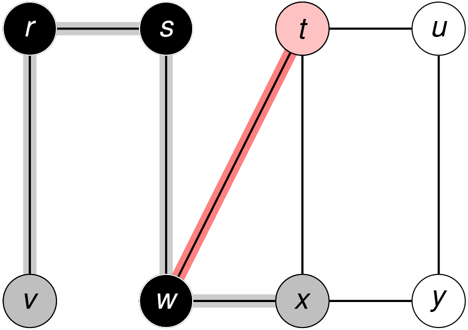
t	x	v

Example

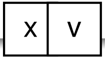


Queue

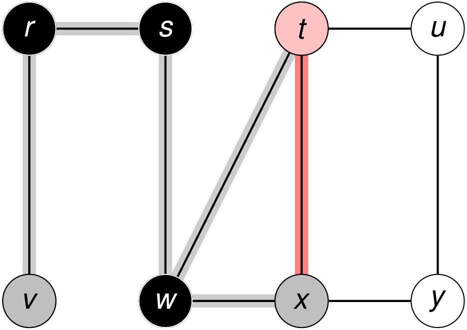
Example



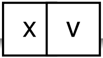
Queue



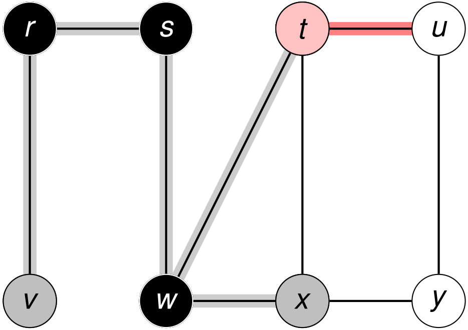
Example



Queue



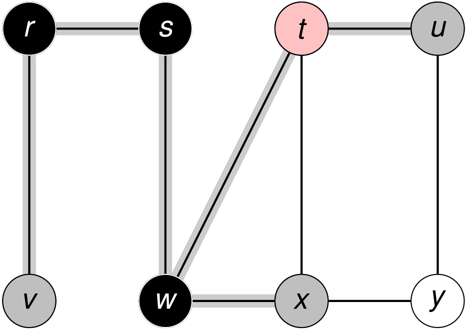
Example



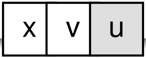
Queue

x	v

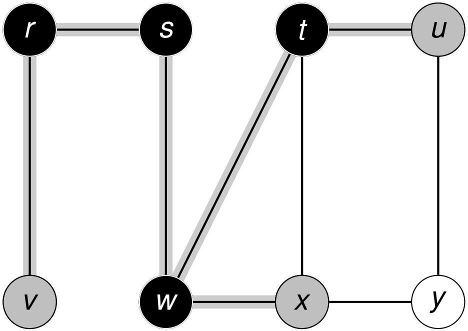
Example



Queue



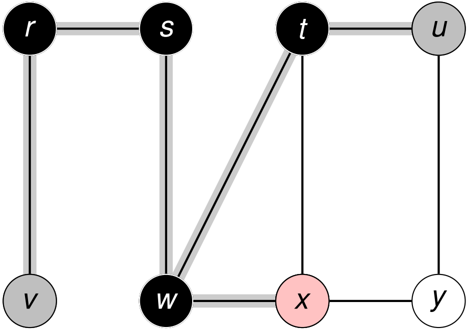
Example



Queue

x	v	u

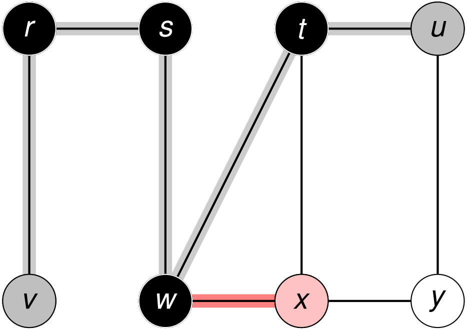
Example



Queue

x	v	u

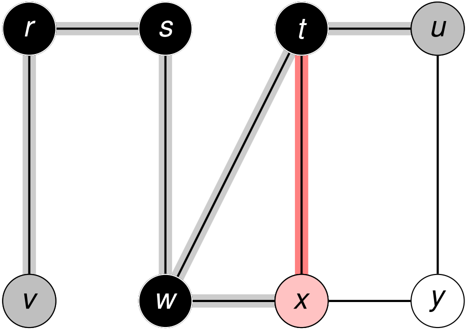
Example



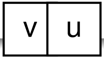
Queue

v	u

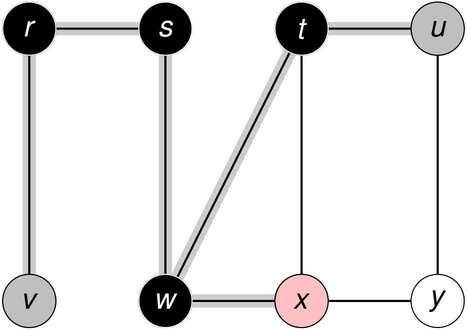
Example



Queue

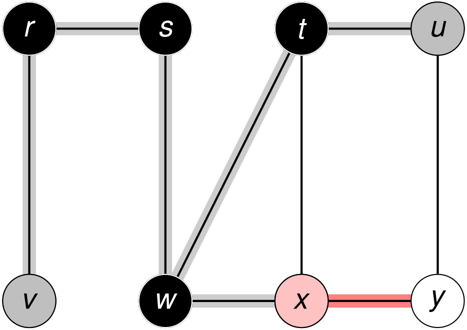


Example

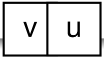


Queue

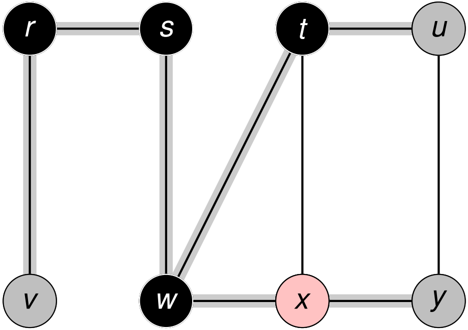
Example



Queue



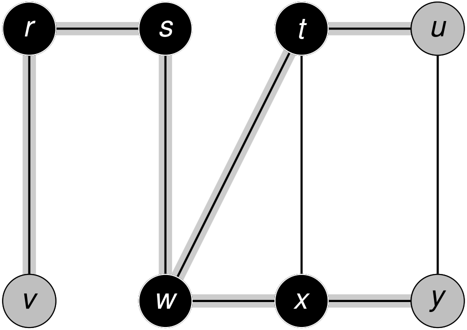
Example



Queue

v	u	y

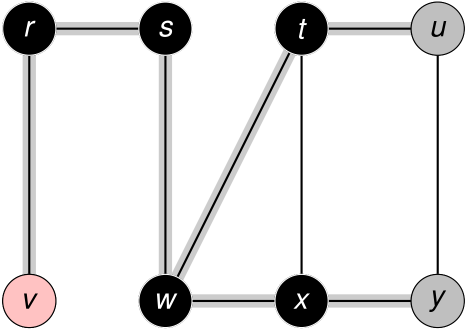
Example



Queue

v	u	y

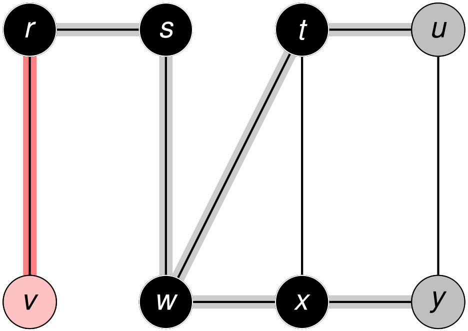
Example



Queue

v	u	y

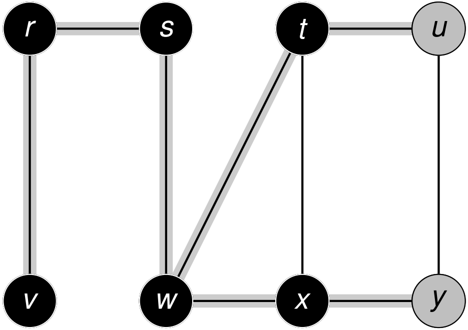
Example



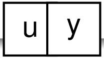
Queue

u	y

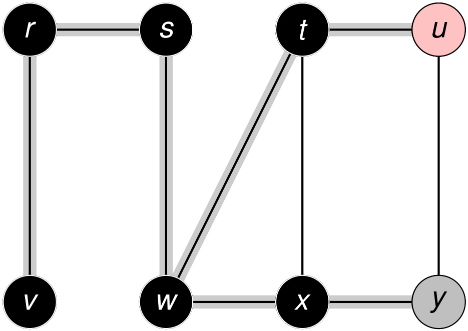
Example



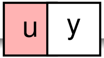
Queue



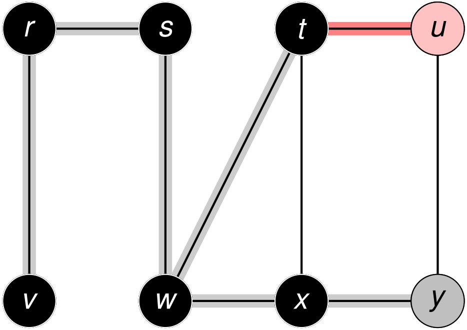
Example



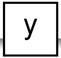
Queue



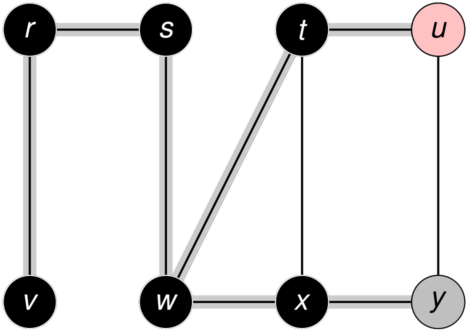
Example



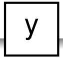
Queue



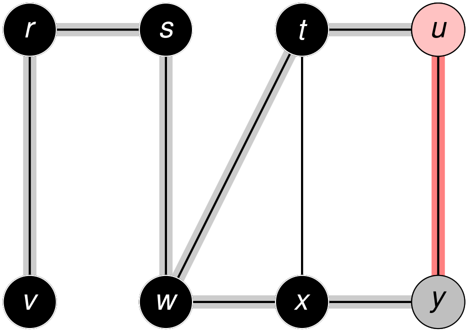
Example



Queue

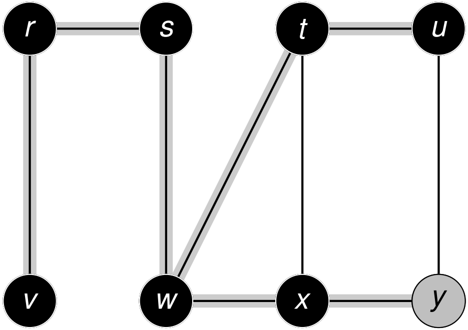


Example



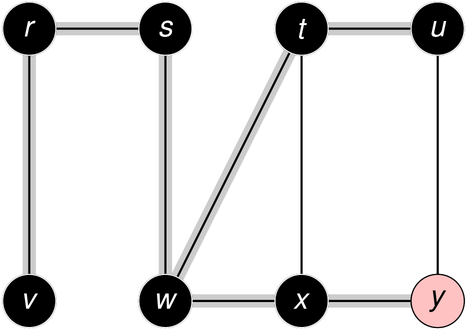
Queue

Example

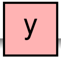


Queue

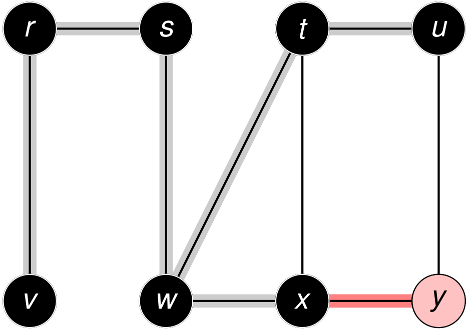
Example



Queue

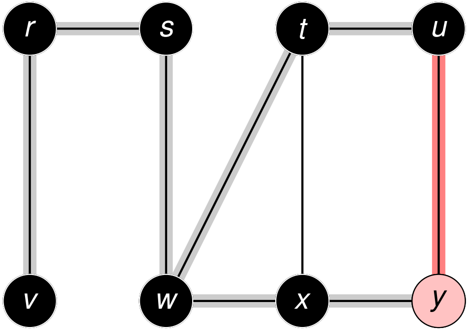


Example



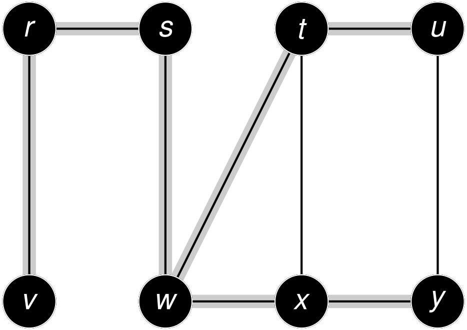
Queue

Example



Queue

Example



Queue

Analysis

Algorithm 2 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do
        $C_u \leftarrow' White'$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow `Gray$
▷ Add v to the queue
                $Enqueue(Q, v)$
            end if
        end for
▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$
    end while
end procedure
```

Analysis

Algorithm 3 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do
        $C_u \leftarrow' White'$                  } $\Theta(n)$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow `Gray$
  ▷ Add v to the queue
                $Enqueue(Q, v)$
            end if
        end for
  ▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$
    end while
end procedure
```

Analysis

Algorithm 4 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do                    $\Theta(n)$
        $C_u \leftarrow' White'$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow$ '$Gray$
▷ Add v to the queue                                 $\Theta(1)$
                $Enqueue(Q, v)$
            end if
        end for
▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$
    end while
end procedure
```

Analysis

Algorithm 5 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do
        $C_u \leftarrow' White'$                        $\Theta(n)$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow `Gray$
  ▷ Add v to the queue                                  $\Theta(1)$      $\Theta(Deg(u))$
                $Enqueue(Q, v)$
            end if
        end for
  ▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$
    end while
end procedure
```

Analysis

Algorithm 6 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do
        $C_u \leftarrow' White'$                        } $\Theta(n)$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow `Gray$
    ▷ Add v to the queue                                } $\Theta(1)$   } $\Theta(Deg(u))$
                $Enqueue(Q, v)$
            end if
        end for
    ▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$                        } $\Theta(1)$
    end while
end procedure
```

Analysis

Algorithm 7 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do                          }
        $C_u \leftarrow' White'$                           }  $\Theta(n)$
    end for                                                }
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then                        }
                $C_v \leftarrow$ '$Gray$                   }
                ▷ Add v to the queue                       }  $\Theta(1)$   }  $\Theta(Deg(u))$   }  $\Sigma_{\forall u \in Q}\Theta(Deg(u)) +$
                $Enqueue(Q, v)$                            }                }                     }  $\Sigma_{\forall u \in Q}\Theta(1)$
            end if                                         }                }                     }
        end for                                                             }                     }
        ▷ Remove head of the queue                                                                }
        $Dequeue(Q)$                                                        }                     }
        $C_u \leftarrow' Black'$                                            }  $\Theta(1)$        }
    end while                                                                                     }
end procedure
```

Analysis

Algorithm 8 BFS Pseudocode

```
procedure BFS($G, s$)
    for each $u \in V - \{s\}$ do
        $C_u \leftarrow' White'$
    end for
    $C_s \leftarrow' Gray'$
    $Enqueue(Q, s)$   *** Q is FIFO Queue ***
    while $Q \neq \Phi$ do
        $u \leftarrow head(Q)$
        for each $v \in Adj_u$ do
            if $C_v =' White'$ then
                $C_v \leftarrow$ '$Gray$
                ▷ Add v to the queue
                $Enqueue(Q, v)$
            end if
        end for
        ▷ Remove head of the queue
        $Dequeue(Q)$
        $C_u \leftarrow' Black'$
    end while
end procedure
```

$\Theta(n)$

$\Theta(1)$

$\Theta(Deg(u))$

$\Theta(1)$

$\Sigma_{\forall u \in Q}\Theta(Deg(u)) +$
$\Sigma_{\forall u \in Q}\Theta(1)$

Running Time

Initialisation

$O(n)$

Queuing/Dequeuing

Each $u \in V$ enqueued/dequeued exactly once ($\forall u \in Q$ is same as $\forall u \in V$) $\Sigma_{\forall u \in V} O(1) = O(n)$

Checking Adjacent nodes

$\Sigma_{\forall u \in V} O(Deg(u)) = O(\Sigma_{\forall u \in V}(Deg(u))) = O(m)$

Total

$O(n) + O(m) + O(n) = O(m + n)$

Applications

- Shortest distance (How?)
- Finding Cycle (How?)
- Connected Components in Undirected Graph (How?)

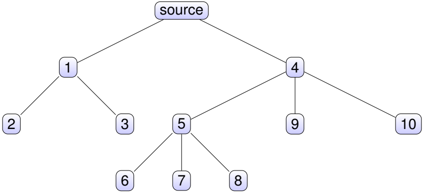
Outline

1. Introduction
    - Representation
    - Graph Traversal
2. BFS
3. DFS
    - Application: Topological Sort
    - Application: Strongly Connected Components

Introduction

Visiting Order

Levels: Go to the deepest level; Backtrack when no other option.



Data Structure used

Stack(LIFO)

- Explicit
- Implicit (Recursive)

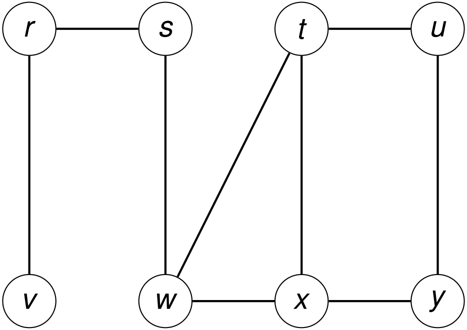
Algorithm(recursive)

Algorithm 9 DFS (Basic Algo)

```
procedure BasicDFS($G$)   *** $G = (V, E)$ ***
    Initialize all nodes as White(unvisited)
    for each $u \in V$ do
        if $u$ is unvisted(White) then
            BasicDFS-visit($u$)
        end if
    end for
end procedure

procedure BasicDFS-visit($u$)
    Mark $u$ as Gray(visited)
    for each $v$ adjacent to $u$ do
        if $v$ is unvisted(White) then
            BasicDFS-visit($v$)
        end if
    end for
    Mark $u$ as Black(dead)
end procedure
```

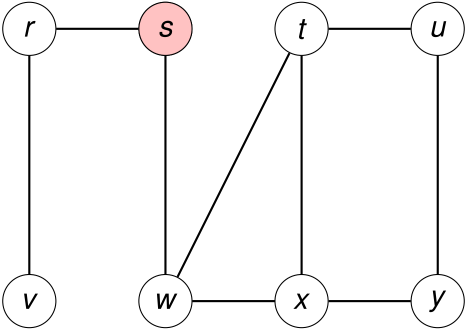
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

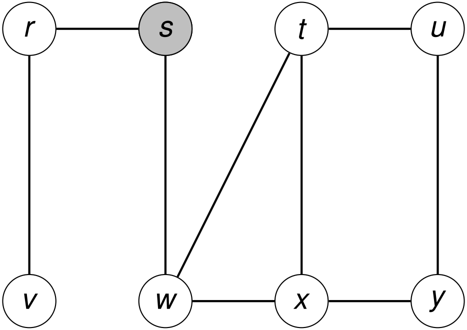
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

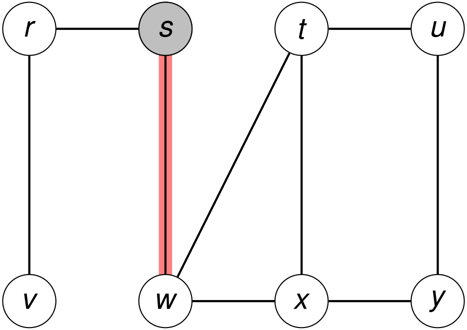
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

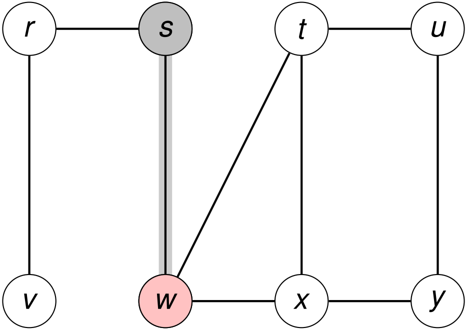
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

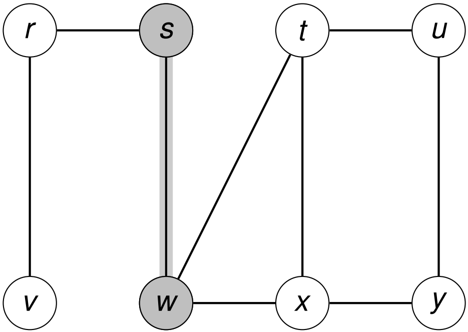
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

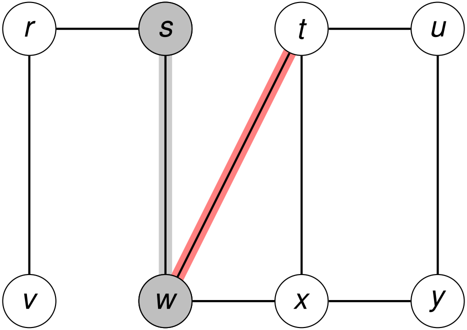
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

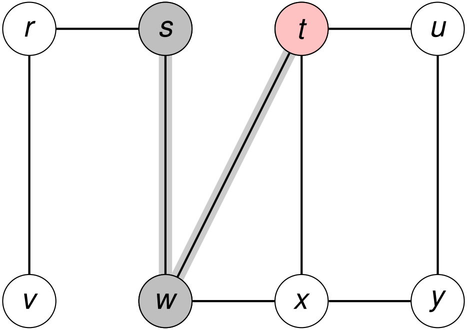
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

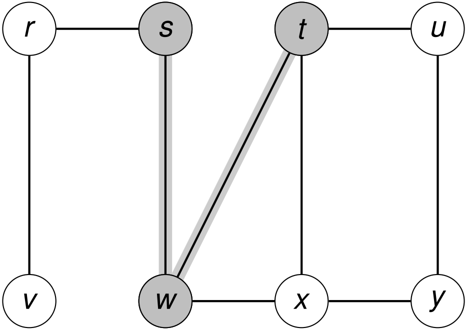
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

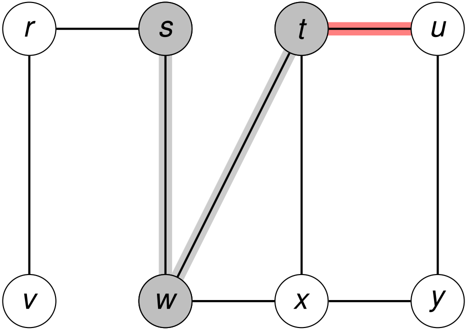
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

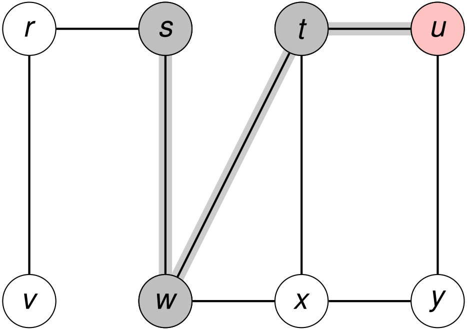
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

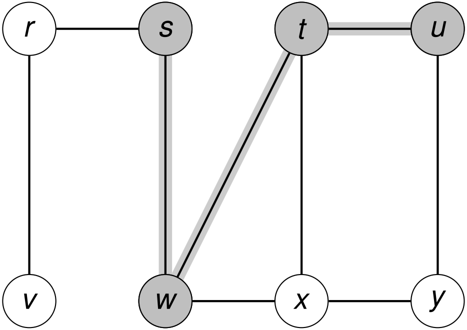
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

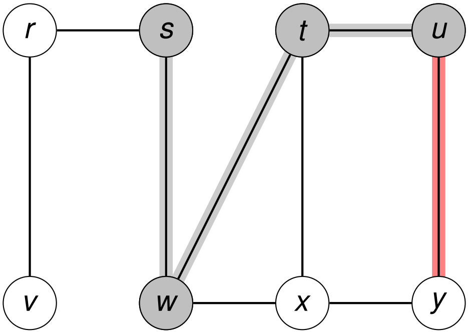
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

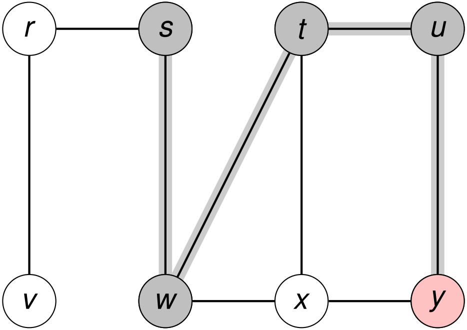
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

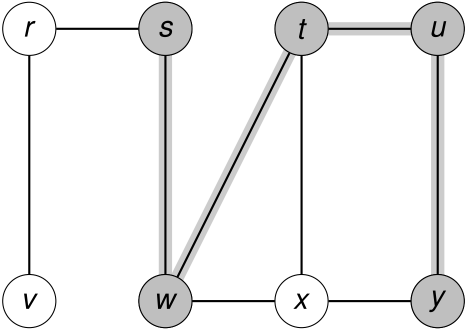
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

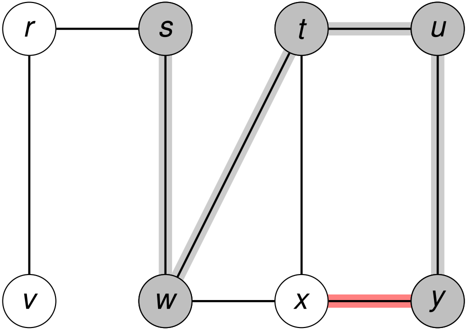
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

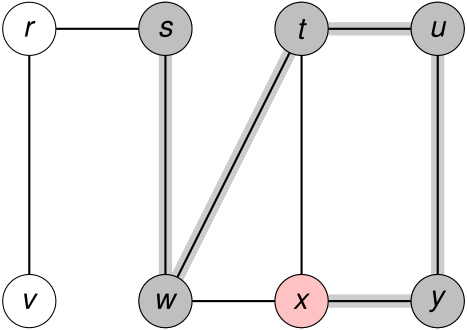
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

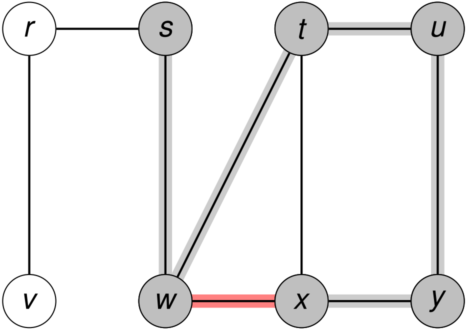
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

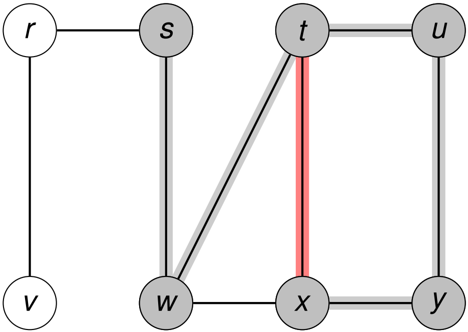
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

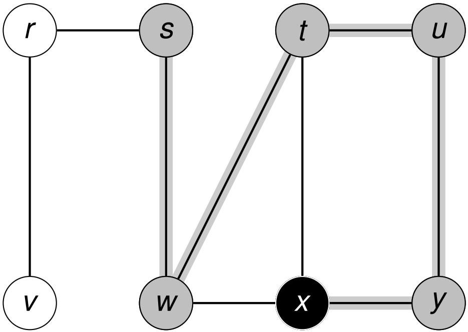
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

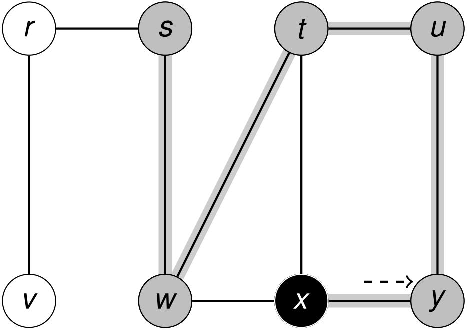
Example



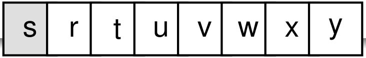
Vertex Set (V)

s	r	t	u	v	w	x	y

Example

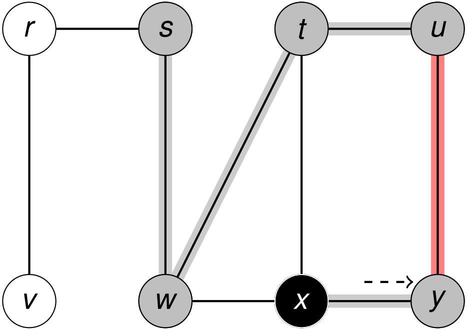


Vertex Set (V)



s	r	t	u	v	w	x	y

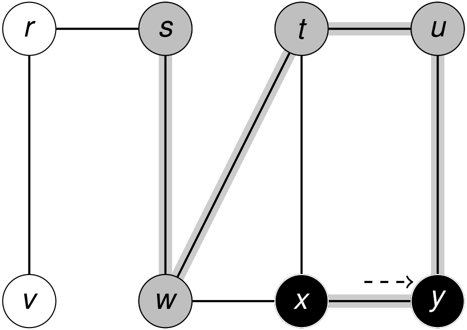
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

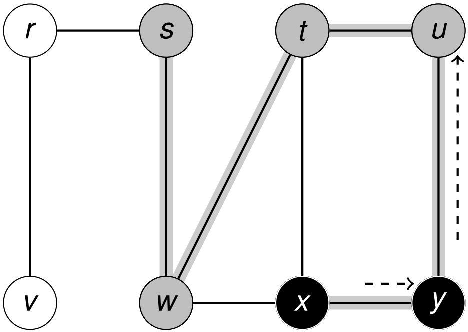
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

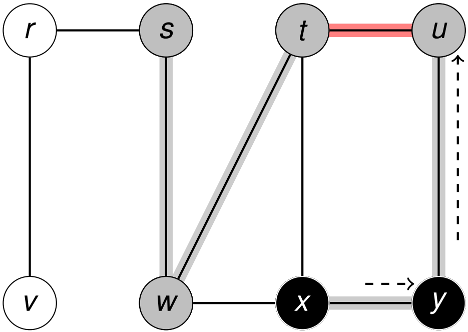
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

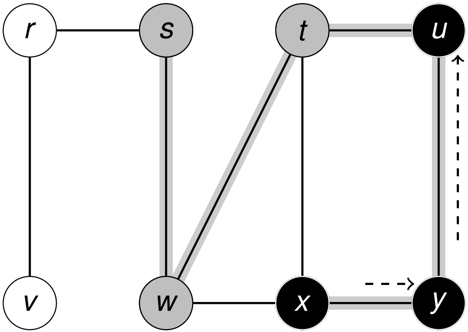
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

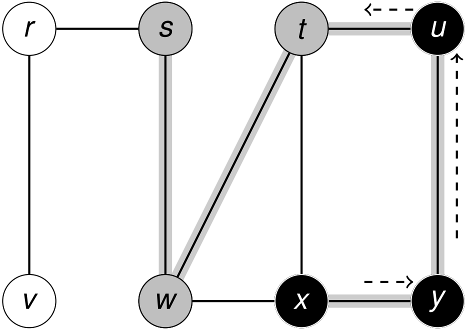
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

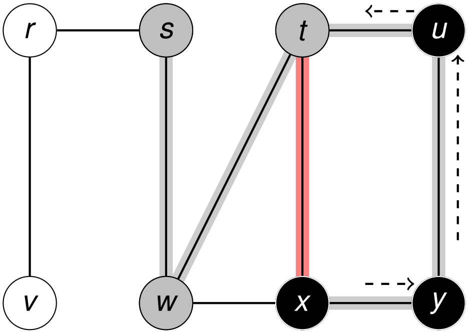
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

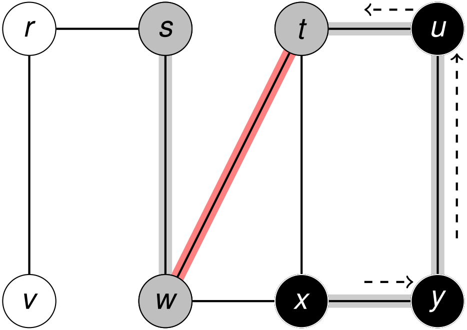
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

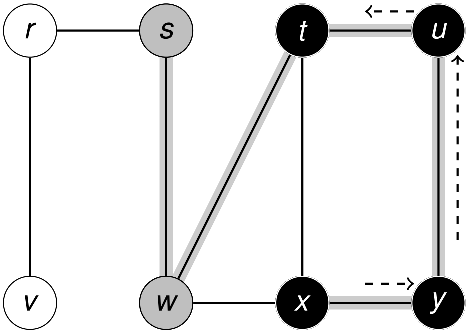
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

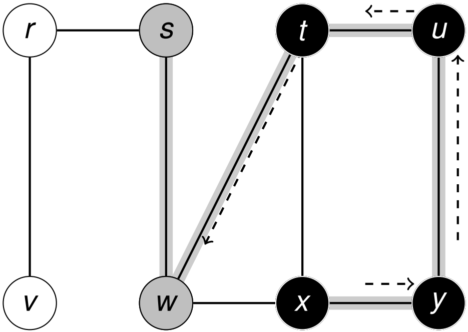
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

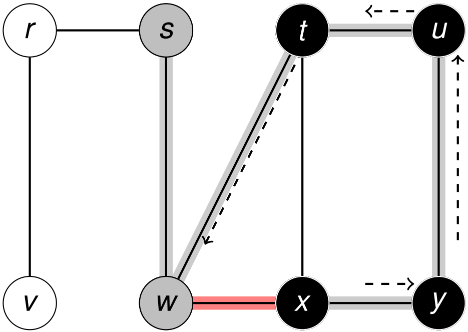
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

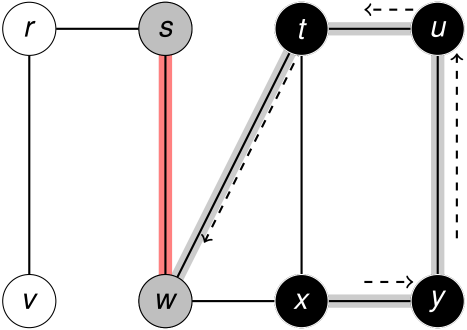
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

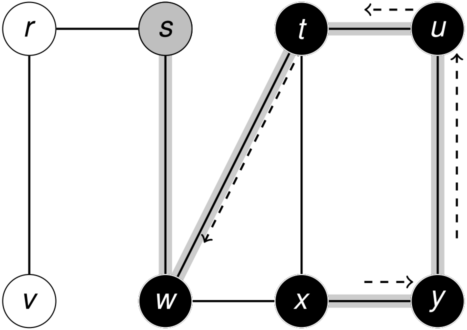
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

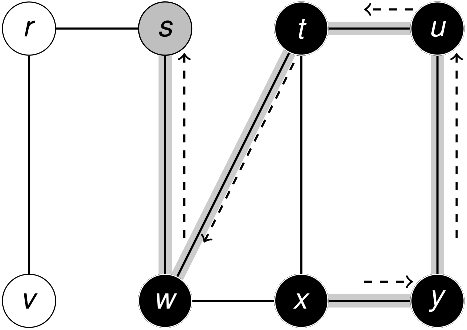
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

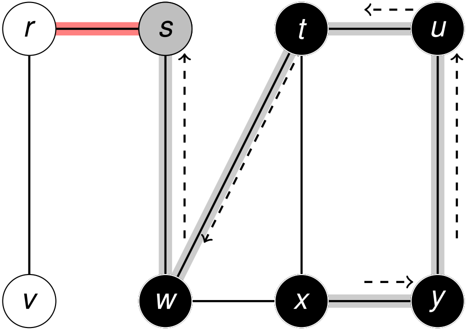
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

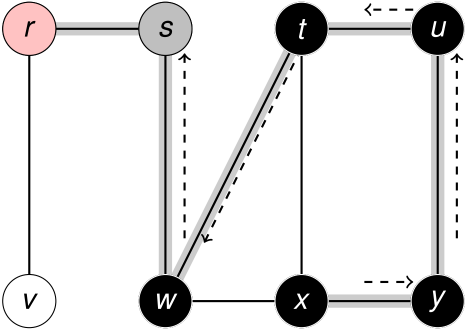
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

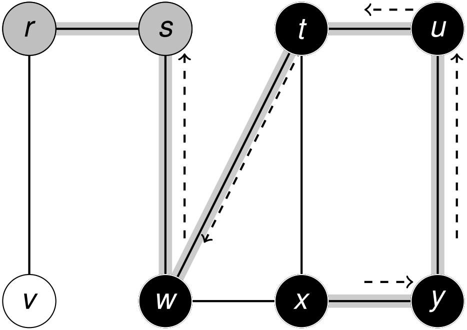
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

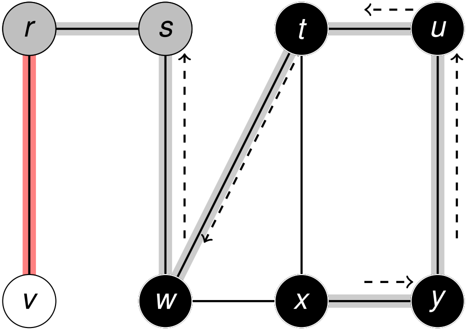
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

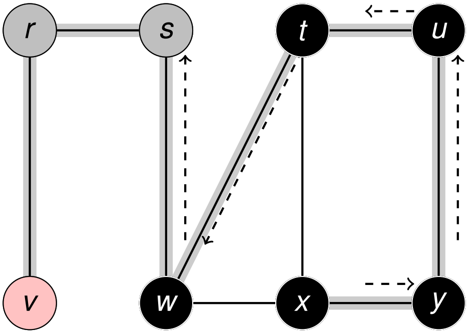
Example



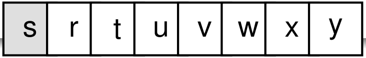
Vertex Set (V)

s	r	t	u	v	w	x	y

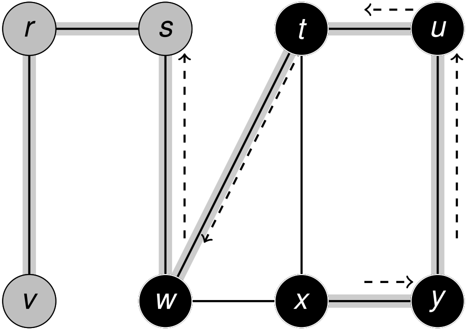
Example



Vertex Set (V)



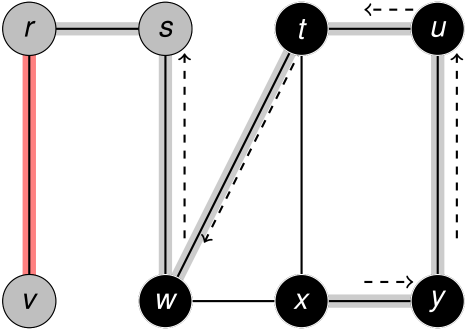
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

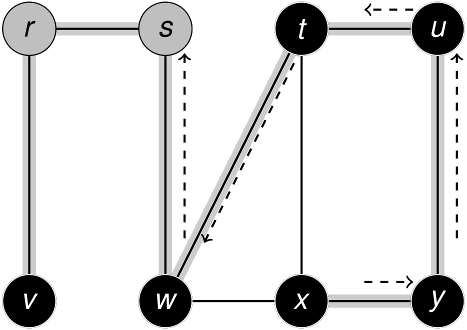
Example



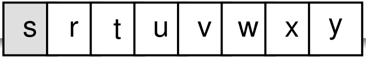
Vertex Set (V)

s	r	t	u	v	w	x	y

Example

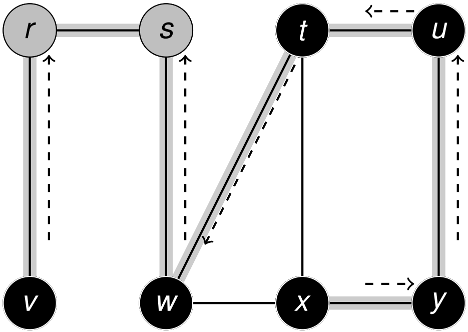


Vertex Set (V)



s	r	t	u	v	w	x	y

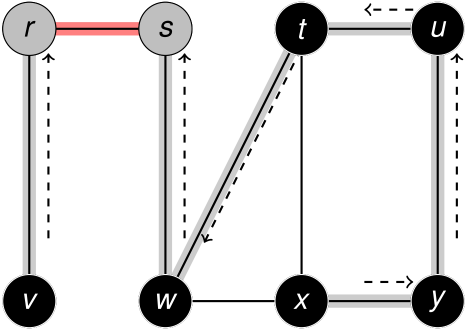
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

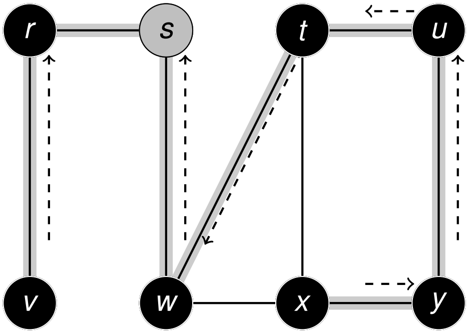
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

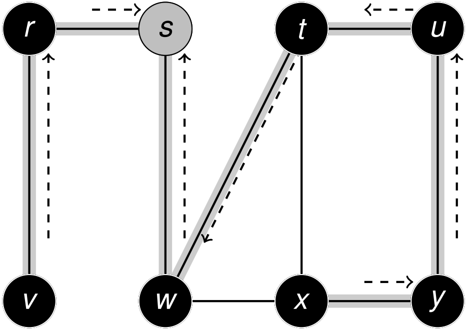
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

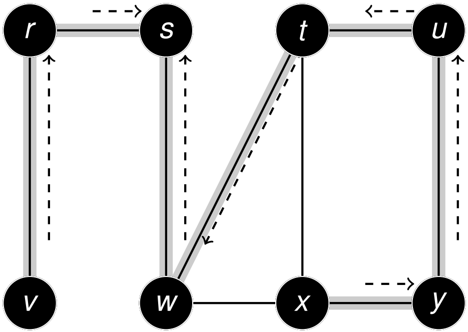
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

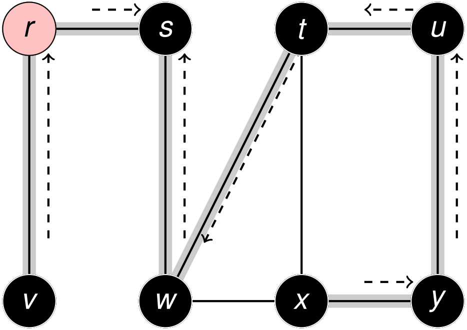
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

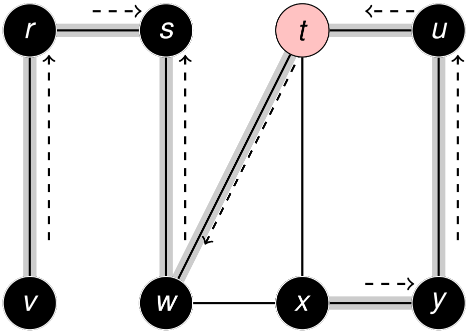
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

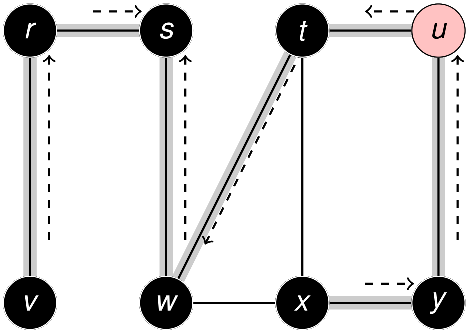
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

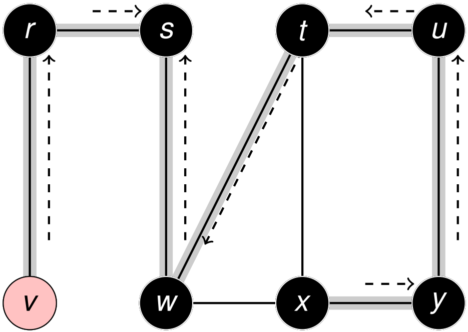
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

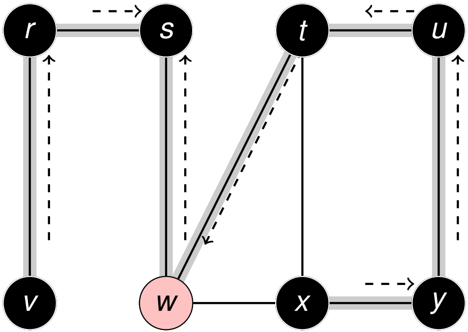
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

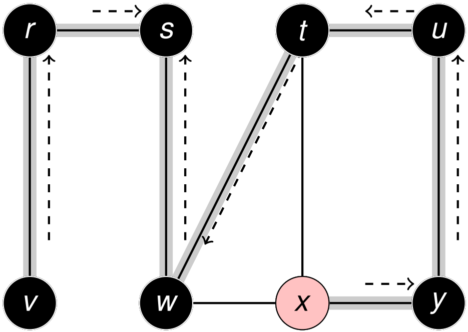
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

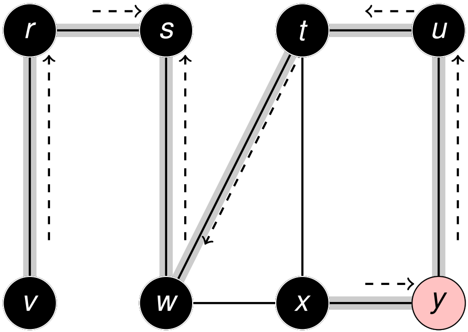
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

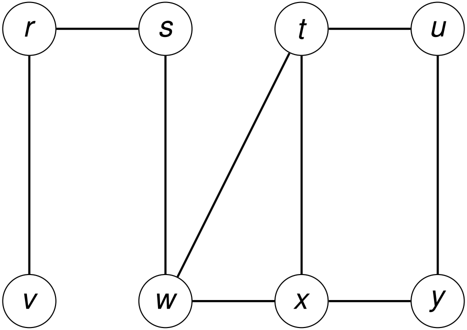
Example



Vertex Set (V)

s	r	t	u	v	w	x	y

Example



Vertex Set (V)

Analysis

Algorithm 10 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 11 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                    } $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 12 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                         $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)                           $\Sigma_{allu \in V} time_{DFS-visit(u)}$
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 13 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                    }  $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)                          }  $\Sigma_{all u \in V} time_{DFS-visit(u)}$
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)                          }  $\Theta(1)$
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 14 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                          $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)                                  $\sum_{all u \in V} time_{DFS-visit(u)}$
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)                                  $\Theta(1)$   $\Theta(Deg(u))$
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 15 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                      } $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)                          } $\Sigma_{all u \in V} time_{DFS-visit(u)}$
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)                          } $\Theta(1)$ } $\Theta(Deg(u))$
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$                                      } $\Theta(1)$
    $f_u \leftarrow t$
end procedure
```

Analysis

Algorithm 16 DFS

```
procedure DFS($G$)   *** $G = (V, E)$ ***
    for each $u \in V$ do
        $C_u \leftarrow$ '$White$'                    } $\Theta(n)$
    end for
    $t \leftarrow 0$
    for each $u \in V$ do
        if $C_u =$ '$White$' then
            DFS-VISIT($u$)                            } $\Sigma_{all u \in V} time_{DFS-visit(u)}$
        end if
    end for
end procedure

procedure DFS-VISIT($u$)
    $C_u \leftarrow$ '$Gray$
    $t \leftarrow t + 1$
    $d_u \leftarrow t$
    for each $v \in Adj_u$ do
        if $C_v =$ '$White$' then
            DFS-VISIT($v$)                            } $\Theta(1)$ } $\Theta(Deg(u))$ | $\Theta(Deg(u)) +$ $\Theta(1)$
        end if
    end for
    $C_u \leftarrow$ '$Black$'
    $t \leftarrow t + 1$                              } $\Theta(1)$
    $f_u \leftarrow t$
end procedure
```

Running Time

Initialisation

$O(n)$

DFS-visit for a node u

$O(Deg(u)) + O(1)$

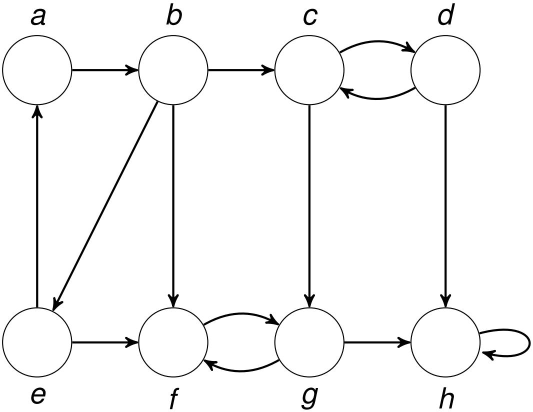
DFS-visit for all nodes

$$\Sigma_{\forall u \in V}(O(Deg(u)) + O(1)) = O(\Sigma_{\forall u \in V}(Deg(u) + O(1)))$$
$$= O(\Sigma_{\forall u \in V}(Deg(u)) + (\Sigma_{\forall u \in V}(O(1)))$$
$$= O(2m(\text{or } m) + n) = O(m + n)$$

Total

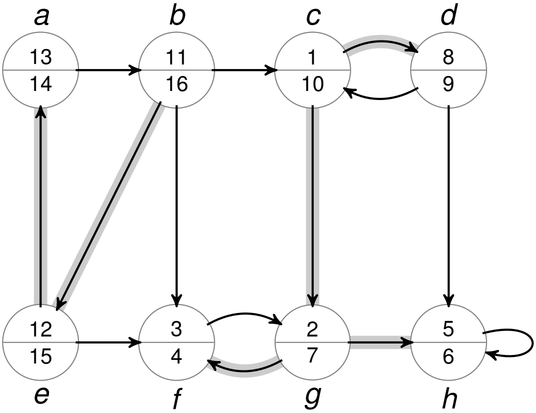
$O(m + n) + O(n) = O(m + n)$

Example 2



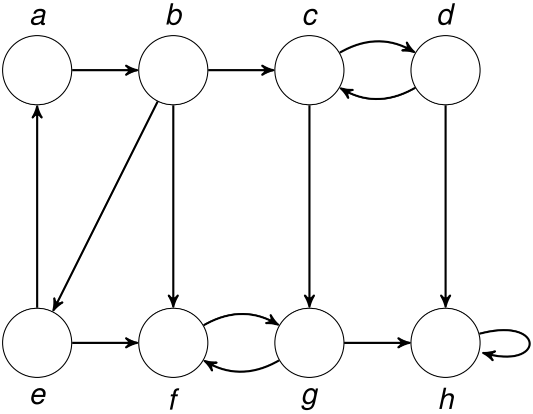
$$V = \{c, d, b, e, a, f, g\}$$

Example 2



$$V = \{c, d, b, e, a, f, g\}$$

Example 2



$$V = \{c, d, b, e, a, f, g\}$$

Does the number of DFS trees in a DFS forest depend on the order of vertices (chosen for traversal)?

Applications

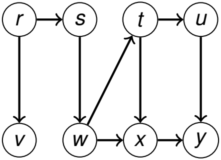
- Finding Cycle (How?)
- Topological sorting
- Strongly Connected Components in Directed Graph

Topological Sort

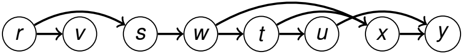
Definition

A *topological sort of a DAG* $G = V, E$ *is a linear ordering of all its vertices such that if* G *contains an edge* (u, v)*, then* u *apperas before* v *in the ordering.*

Example

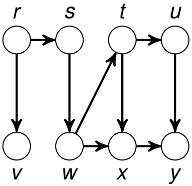


Topological order:



Algorithm

Example



Pseudocode

Algorithm 17 Topological Sort

```
procedure TOPOLOGICAL-SORT($G$)    *** $G = (V, E)$ ***
    Call DFS($G$) to compute $f_u \forall u \in V$
    As each vertex finishes, insert it at the front of a linked list
    return the linked list of vertices
end procedure
```

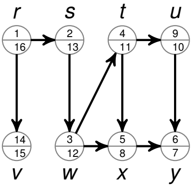
Algorithm

Pseudocode

Algorithm 18 Topological Sort

```
procedure TOPOLOGICAL-SORT(G)   *** $G = (V, E)$ ***
    Call DFS(G) to compute $f_u \forall u \in V$
    As each vertex finishes, insert it at the front of a linked list
    return the linked list of vertices
end procedure
```

Example



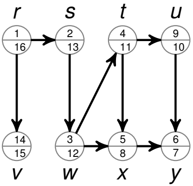
Algorithm

Pseudocode

Algorithm 19 Topological Sort

```
procedure TOPOLOGICAL-SORT($G$)   *** $G = (V, E)$ ***
    Call DFS($G$) to compute $f_u \forall u \in V$
    As each vertex finishes, insert it at the front of a linked list
    return the linked list of vertices
end procedure
```

Example



Vertices in decreasing order of the finishing time $=$ $r(16), v(15), s(13), w(12), t(11), u(10), x(8), y(7)$

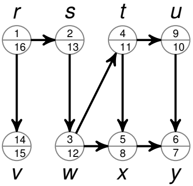
Algorithm

Pseudocode

Algorithm 20 Topological Sort

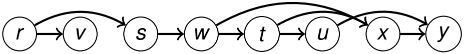
```
procedure TOPOLOGICAL-SORT($G$)   *** $G = (V, E)$ ***
    Call DFS($G$) to compute $f_u \forall u \in V$
    As each vertex finishes, insert it at the front of a linked list
    return the linked list of vertices
end procedure
```

Example

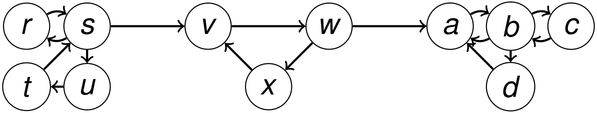


Vertices in decreasing order of the finishing time $=$ $r(16), v(15), s(13), w(12), t(11), u(10), x(8), y(7)$

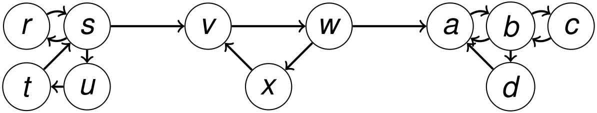
Topological order:



DFS and SCCs

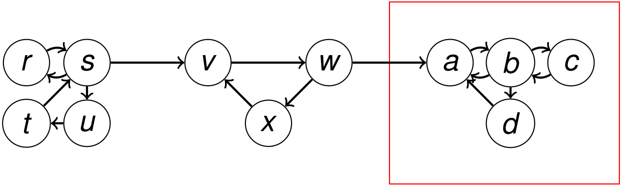


DFS and SCCs



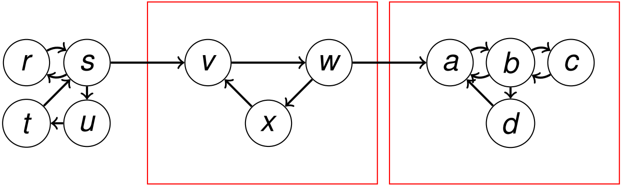
Let V = $= \{b, \ldots.., x, \ldots.., s, \ldots.\}$

DFS and SCCs



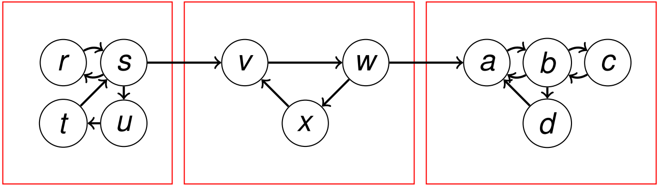
Let V = $= \{b, ..., x, ..., s, ...\}$

DFS and SCCs



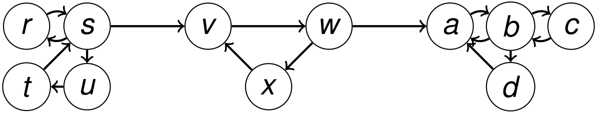
Let V = $= \{b, \ldots.., x, \ldots.., s, \ldots.\}$

DFS and SCCs



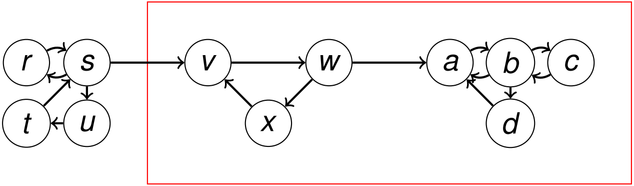
Let V = $= \{b, ..., x, ..., s, ...\}$

DFS and SCCs



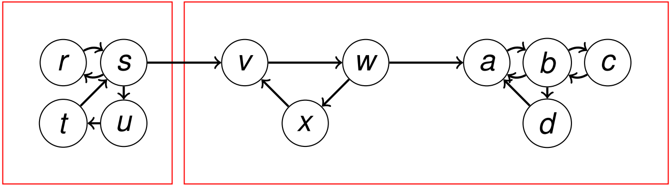
Let V = $= \{x, ..., s, ...\}$

DFS and SCCs



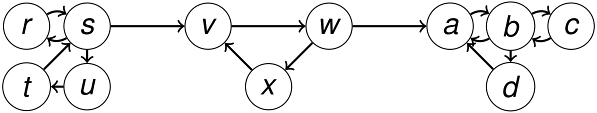
Let V = $= \{x, ..., s, ...\}$

DFS and SCCs



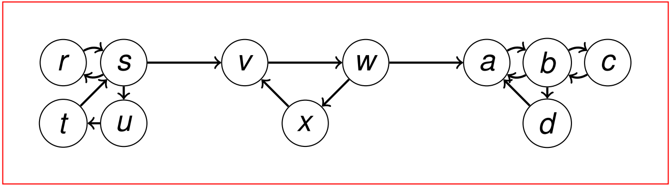
Let V = $= \{x, ..., s, ...\}$

DFS and SCCs



Let V = $= \{s, ...\}$

DFS and SCCs



Let V = $= \{s, ...\}$

Idea behind algorithm to find SCC

Transpose Graph: G^T

- Definition :
  - Informally- Graph obtained after reversing all the edges.
  - $G^T = (V, E^T), E^T = \{(u, v) : (v, u) \in E\}$
- Observation : G and G^T have the same SCCs.

Component Graph: G^{SCC}

- Definition :
  - Informally- Graph obtained after by contracting all edges whose incident vertices are within the same SCCs of G.
  - $G^{SCC} = (V^{SCC}, E^{SCC})$, where V^{SCC} has one vertex for each SCC in G and E^{SCC} has an edge if there's an edge between the corresponding SCC's in G.
- Example :

- Observation: G^{SCC} is a DAG.

Pseudocode

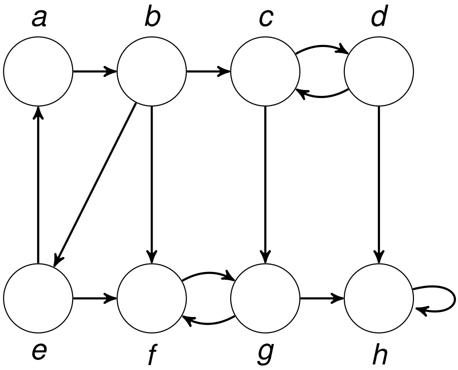
Algorithm 21 Strongly Connected Component

```
procedure FindSCC($G$)   *** $G = (V, E)$ ***
    Call DFS($G$) to compute $f_u \; \forall u \in V$
    Compute $G^T$   *** $G^T = (V, E^T)$, $E^T = \{(u, v) : (v, u) \in E\}$ ***
    Call DFS($G^T$) but considering the vertices in decreasing order of $f_u$ computed in the first call to DFS(above step).
    return the vertices of each tree in the second call to DFS (above step) as a separate SCC.
end procedure
```

Running time: $O(n + m)$

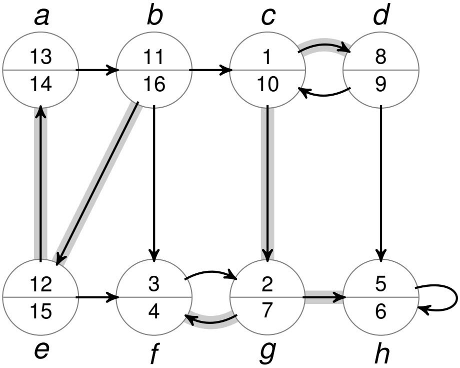
Example

Step1: $DFS(G)$



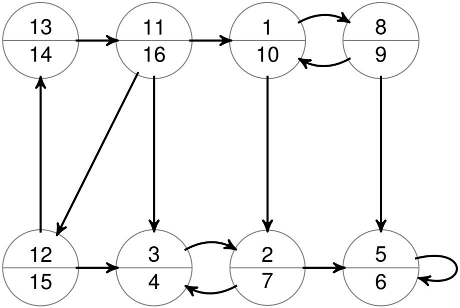
Example

Step1: $DFS(G)$



Example

Step1: *DFS*(*G*)



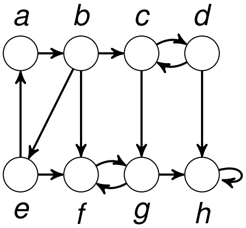
Vertices in decreasing order of the finishing time $= b(16), e(15), a(14), c(10), d(9), g(7), h(6), f(4)$

Example

Step2: Compute G^T

$G = (V, E)$

$G^T = (V, E^T)$, $E^T = \{(u, v) : (v, u) \in E\}$

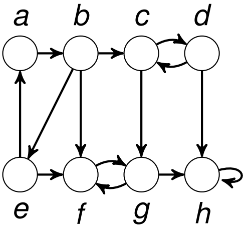


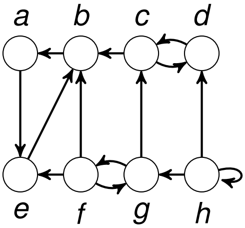
Example

Step2: Compute G^T

$G = (V, E)$

$G^T = (V, E^T)$, $E^T = \{(u, v) : (v, u) \in E\}$

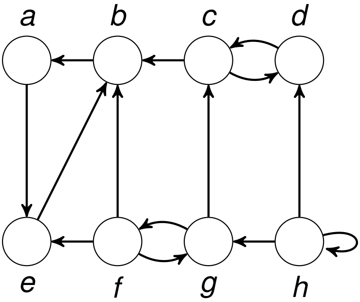




Example

Step3: $DFS(G^T)$ in decreasing order of f_u from step 1

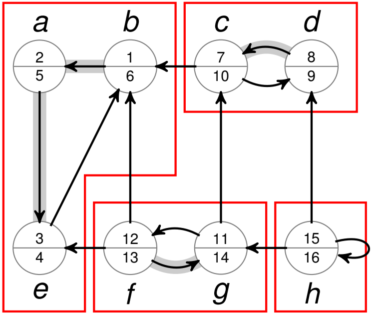
Vertices in decreasing order of the finishing time $= b, e, a, c, d, g, h, f$



Example

Step3: $DFS(G^T)$ in decreasing order of f_u from step 1

Vertices in decreasing order of the finishing time $= b, e, a, c, d, g, h, f$



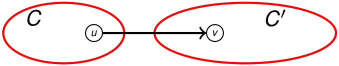
Why the algorithm works?

Idea

By considering vertices in second DFS in decreasing order of finishing times (obtained from first DFS), vertices of the component graph are being visited in topological sort order.

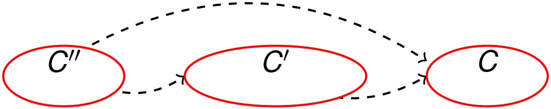
How?

- Notations :
  - d_u, f_u : discovery and finishing time of a vertex u in the first DFS.
  - D_C : Discovery time of the first-discovered vertex in SCC C, i.e.$D_C = min_{u \in C} d_u$.
  - F_C : Finishing time of the last-finished vertex in SCC C, i.e.$D_C = max_{u \in C} f_u$.
- Lemma :Let C and C' be distinct SCCs in $G = (V, E)$. If there is an edge (u, v) in E such that u in C and v in C', then $F_C > F_{C'}$.



- Corollary 1: Let C and C' be distinct SCCs in $G = (V, E)$. If there is an edge (u, v) in E^T where u in C and v in C', then $F_C < F_{C'}$.
- Corollary 2: Let C and C' be distinct SCCs in $G = (V, E)$. If $F_C > F_{C'}$, then there cannot be an edge from C to C' in G^T.

Why the algorithm works? : Intutive Proof

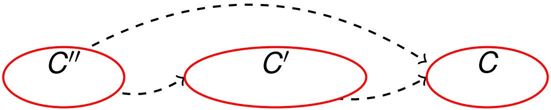


Each time we choose a root for the second DFS, it can reach only

- vertices in its own SCC.
- vertices in SCCs already visited in second DFS.

In effect, vertices of $(G^T)^{SCC}$ in reverse of topologically sorted order.

Why the algorithm works? : Intutive Proof



Each time we choose a root for the second DFS, it can reach only

- vertices in its own SCC.
- vertices in SCCs already visited in second DFS.

In effect, vertices of $(G^T)^{SCC}$ in reverse of topologically sorted order.

What if the first DFS is done on G^T and the second on G?

References

- Cormen, Leiserson, Rivest : *Introduction to Algorithms*