

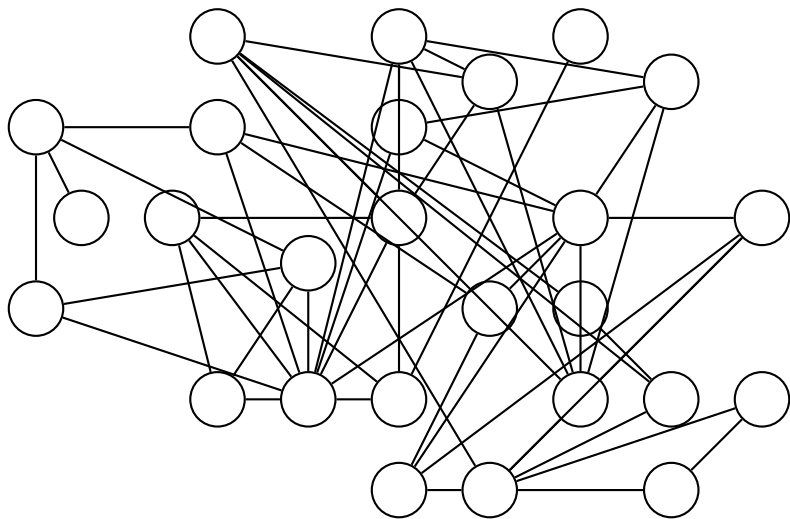
Graphs: Representation and Elementary Algorithms

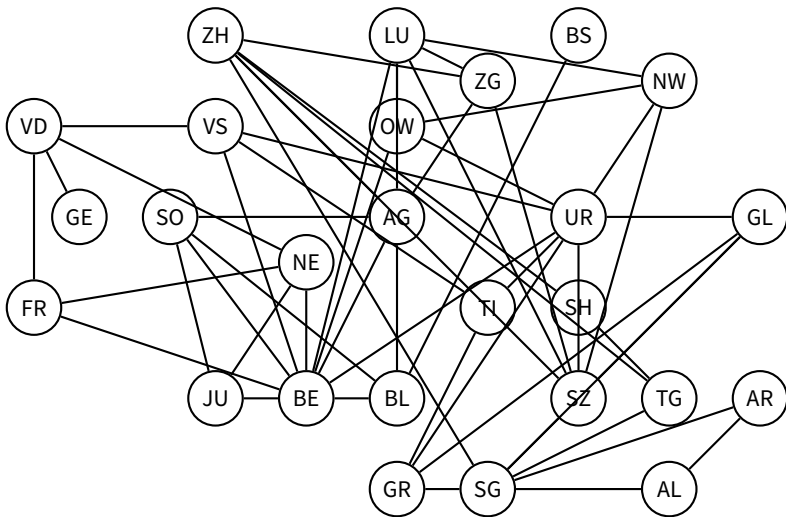
Antonio Carzaniga

Faculty of Informatics
Università della Svizzera italiana

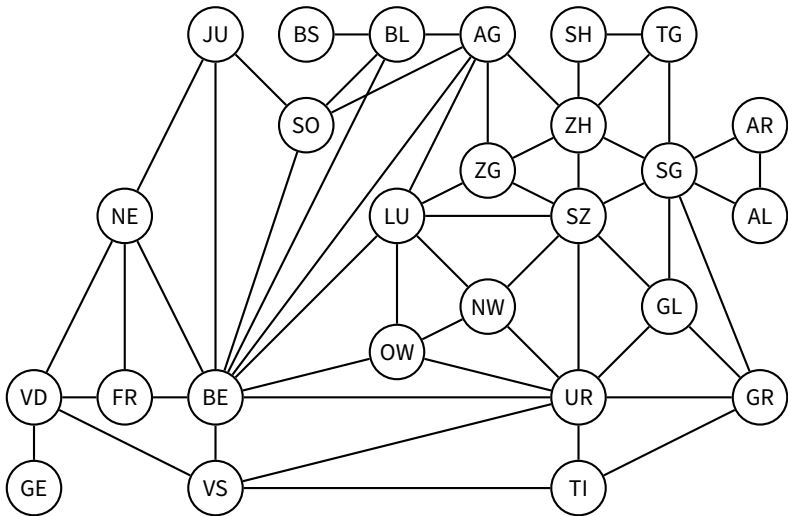
May 6, 2025

- Graphs: definitions
- Representations
- Breadth-first search
- Depth-first search





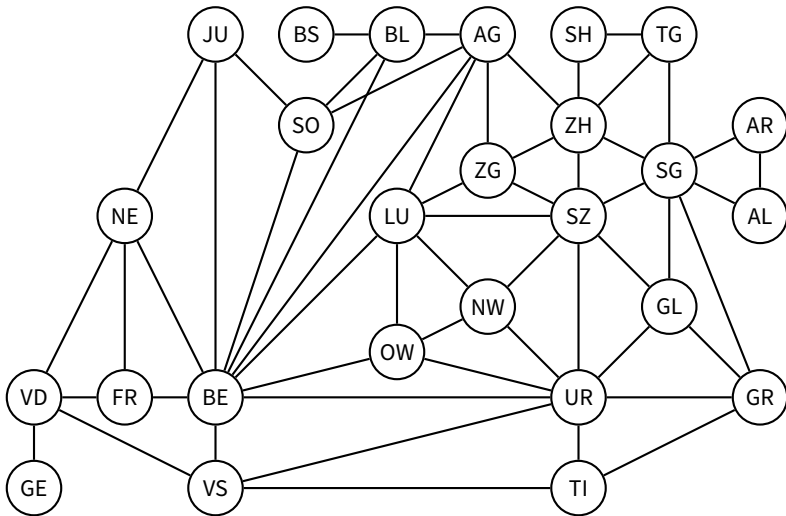
Same Example (Better Layout)



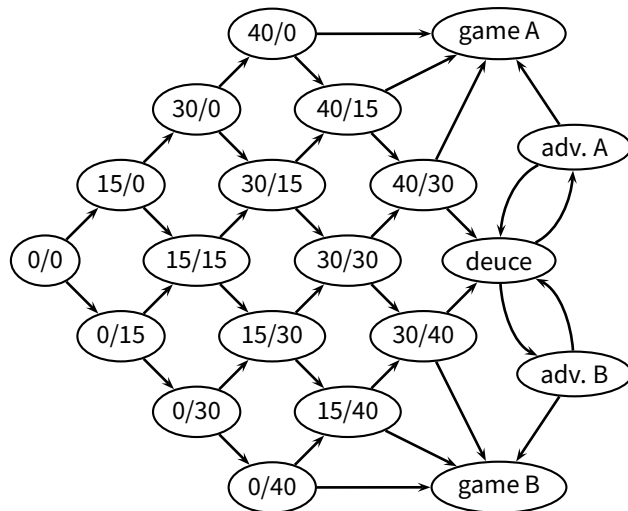
Many Models and Applications

- Social networks: *who knows who*
- The Web graph: *which page links to which*
- The Internet graph: *which router links to which*
- Citation graphs: *who references whose papers*
- Planar graphs: *which country is next to which*
- Well-shaped meshes: *pretty pictures with triangles*
- Geometric graphs: *who is near who*
- Random graphs: *whichever...*

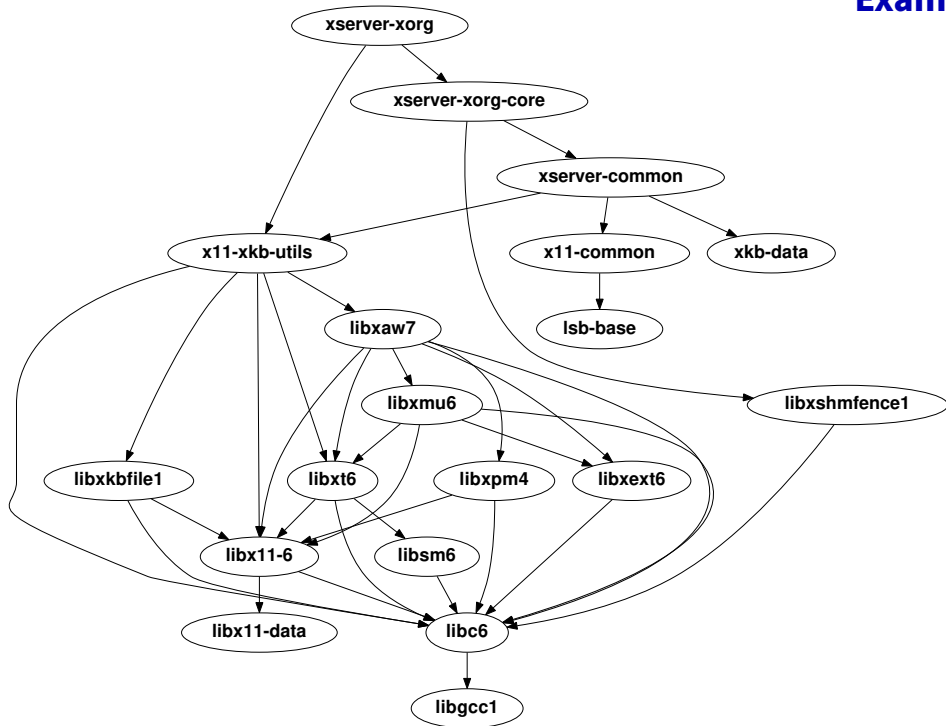
Example (1)



Example (2)



Example (3)



- A *graph*

$$G = (V, E)$$

- V is the set of *vertices* (also called *nodes*)
- E is the set of *edges*

- A **graph**

$$G = (V, E)$$

- V is the set of **vertices** (also called **nodes**)

- E is the set of **edges**

- ▶ $E \subseteq V \times V$, i.e., E is a **relation between vertices**

- ▶ an edge $e = (u, v) \in E$ is a pair of vertices $u \in V$ and $v \in V$

- A **graph**

$$G = (V, E)$$

- V is the set of **vertices** (also called **nodes**)

- E is the set of **edges**

- ▶ $E \subseteq V \times V$, i.e., E is a **relation between vertices**

- ▶ an edge $e = (u, v) \in E$ is a pair of vertices $u \in V$ and $v \in V$

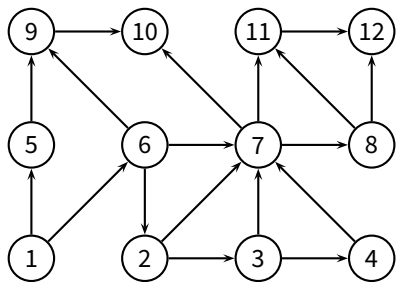
- An **undirected** graph is characterized by a **symmetric** relation between vertices

- ▶ an edge is a set $e = \{u, v\}$ of two vertices

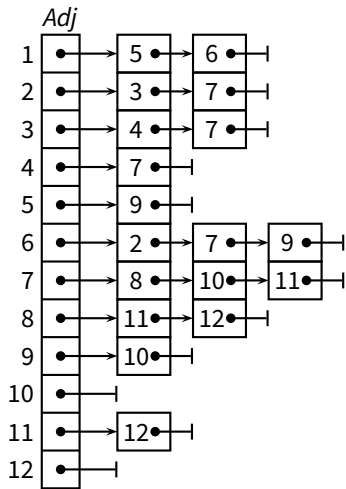
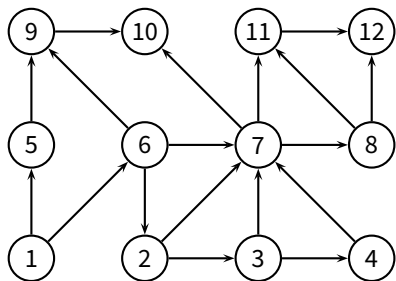
- How do we represent a graph $G = (E, V)$ in a computer?

- How do we represent a graph $G = (E, V)$ in a computer?
- *Adjacency-list representation*
- $V = \{1, 2, \dots, |V|\}$
- G consists of an array Adj
- A vertex $u \in V$ is represented by an element in the array Adj

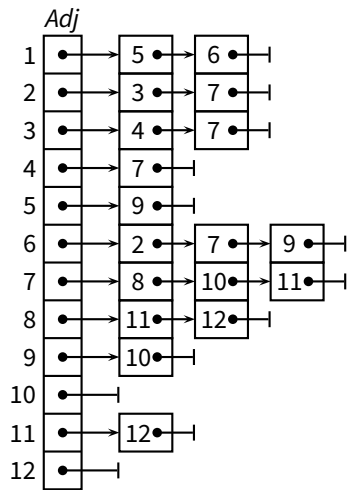
- How do we represent a graph $G = (E, V)$ in a computer?
- *Adjacency-list representation*
- $V = \{1, 2, \dots, |V|\}$
- G consists of an array Adj
- A vertex $u \in V$ is represented by an element in the array Adj
- $Adj[u]$ is the **adjacency list** of vertex u
 - ▶ the list of the vertices that are adjacent to u
 - ▶ i.e., the list of all v such that $(u, v) \in E$



Example

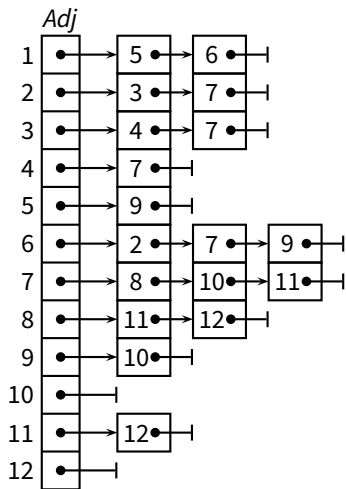


Using the Adjacency List



Using the Adjacency List

- Accessing a vertex u ?

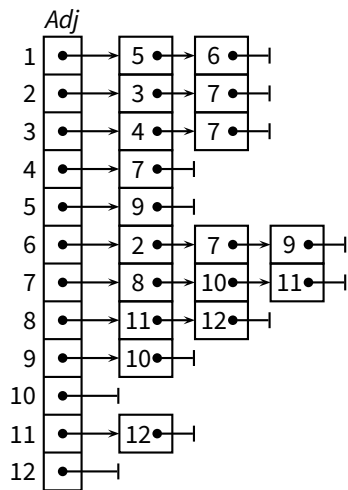


Using the Adjacency List

■ Accessing a vertex u ?

- ▶ optimal

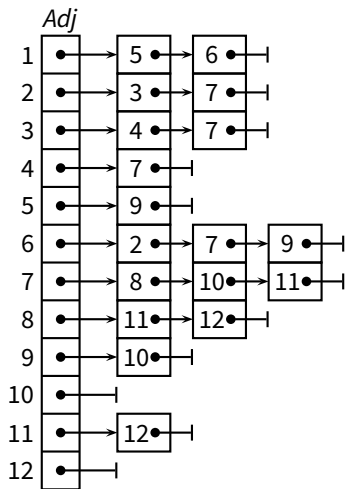
$O(1)$



Using the Adjacency List

- Accessing a vertex u ?
 - ▶ optimal
- Iteration through V ?

$O(1)$



Using the Adjacency List

■ Accessing a vertex u ?

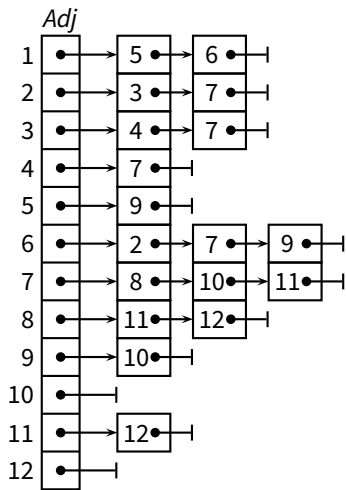
▶ optimal

■ Iteration through V ?

▶ optimal

$O(1)$

$\Theta(|V|)$

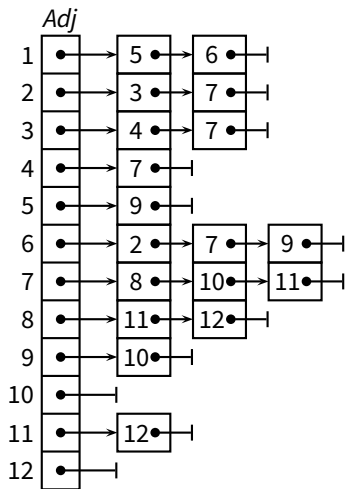


Using the Adjacency List

- Accessing a vertex u ?
 - ▶ optimal
- Iteration through V ?
 - ▶ optimal
- Iteration through E ?

$O(1)$

$\Theta(|V|)$



Using the Adjacency List

■ Accessing a vertex u ?

▶ optimal

■ Iteration through V ?

▶ optimal

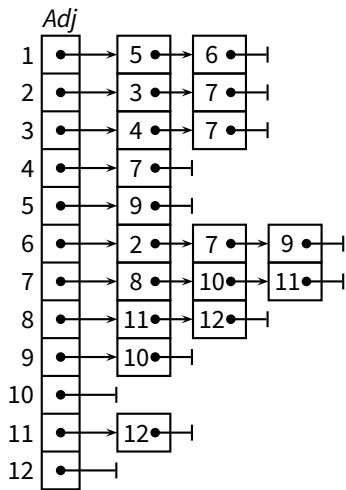
■ Iteration through E ?

▶ okay (not optimal)

$O(1)$

$\Theta(|V|)$

$\Theta(|V| + |E|)$



Using the Adjacency List

■ Accessing a vertex u ?

▶ optimal

■ Iteration through V ?

▶ optimal

■ Iteration through E ?

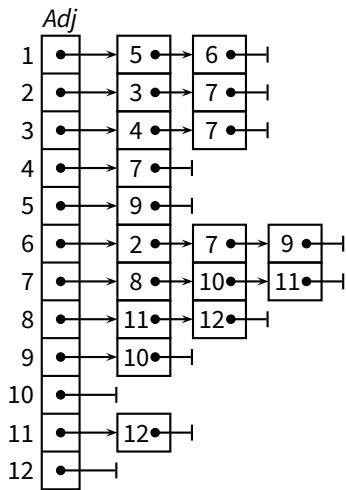
▶ okay (not optimal)

■ Checking $(u, v) \in E$?

$O(1)$

$\Theta(|V|)$

$\Theta(|V| + |E|)$



Using the Adjacency List

■ Accessing a vertex u ?

▶ optimal

■ Iteration through V ?

▶ optimal

■ Iteration through E ?

▶ okay (not optimal)

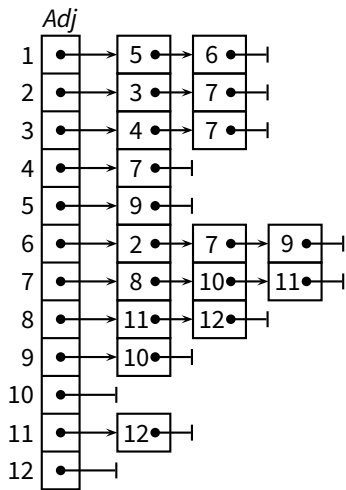
■ Checking $(u, v) \in E$?

$O(1)$

$\Theta(|V|)$

$\Theta(|V| + |E|)$

$O(|V|)$



Using the Adjacency List

■ Accessing a vertex u ?

▶ optimal

■ Iteration through V ?

▶ optimal

■ Iteration through E ?

▶ okay (not optimal)

■ Checking $(u, v) \in E$?

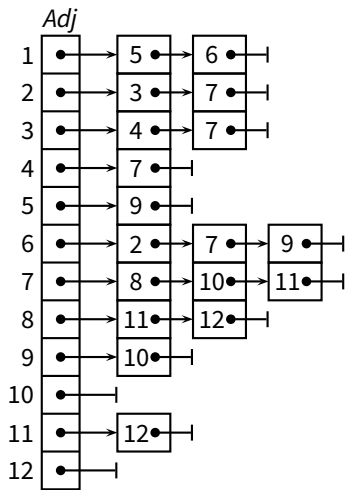
▶ bad

$O(1)$

$\Theta(|V|)$

$\Theta(|V| + |E|)$

$O(|V|)$



Graph Representation (2)

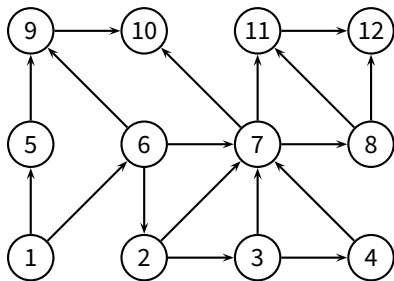
- *Adjacency-matrix representation*

- $V = \{1, 2, \dots, |V|\}$

- G consists of a $|V| \times |V|$ matrix A

- $A = (a_{ij})$ such that

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



- Adjacency-list representation

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

optimal

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

optimal

- Adjacency-matrix representation

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

optimal

- Adjacency-matrix representation

$$\Theta(|V|^2)$$

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

optimal

- Adjacency-matrix representation

$$\Theta(|V|^2)$$

possibly very bad

- Adjacency-list representation

$$\Theta(|V| + |E|)$$

optimal

- Adjacency-matrix representation

$$\Theta(|V|^2)$$

possibly very bad

- When is the adjacency-matrix “very bad”?

Choosing a Graph Representation

■ Adjacency-list representation

- ▶ generally good, especially for its optimal space complexity
- ▶ bad for *dense* graphs and algorithms that require random access to edges
- ▶ preferable for *sparse* graphs or graphs with *low degree*

Choosing a Graph Representation

■ Adjacency-list representation

- ▶ generally good, especially for its optimal space complexity
- ▶ bad for **dense** graphs and algorithms that require random access to edges
- ▶ preferable for **sparse** graphs or graphs with **low degree**

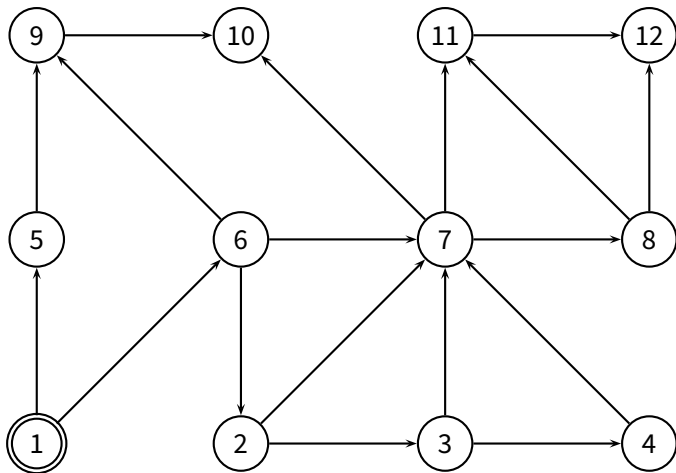
■ Adjacency-matrix representation

- ▶ suffers from a bad space complexity
- ▶ good for algorithms that require random access to edges
- ▶ preferable for **dense** graphs

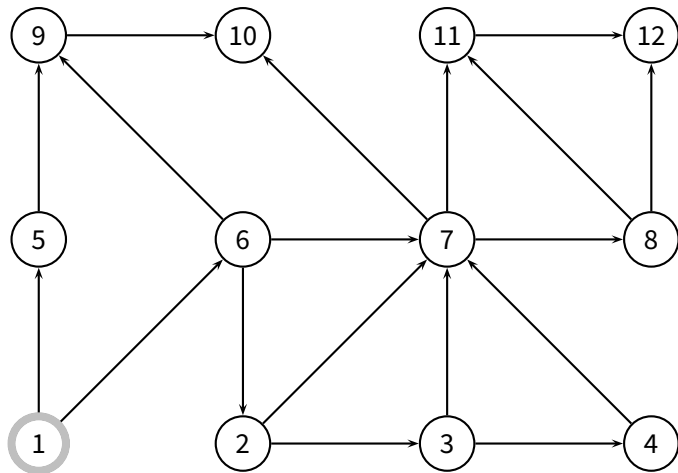
- One of the simplest but also a fundamental algorithm

- One of the simplest but also a fundamental algorithm
- *Input:* $G = (V, E)$ and a vertex $s \in V$
 - ▶ explores the graph, touching all vertices that are reachable from s
 - ▶ iterates through the vertices at increasing distance (edge distance)
 - ▶ computes the distance of each vertex from s
 - ▶ produces a ***breadth-first tree*** rooted at s
 - ▶ works on both *directed* and *undirected* graphs

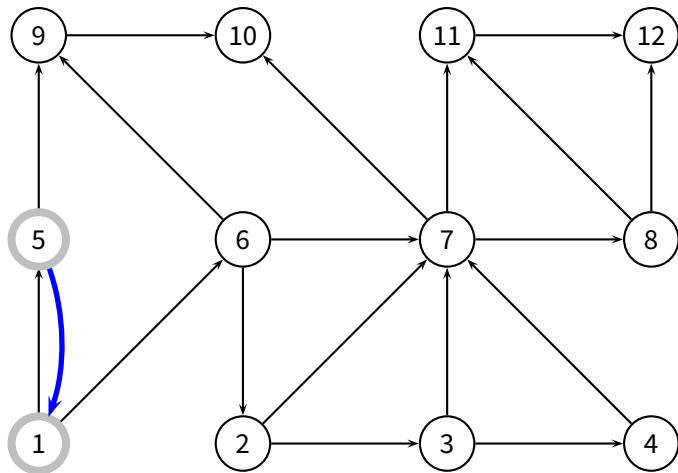
Example

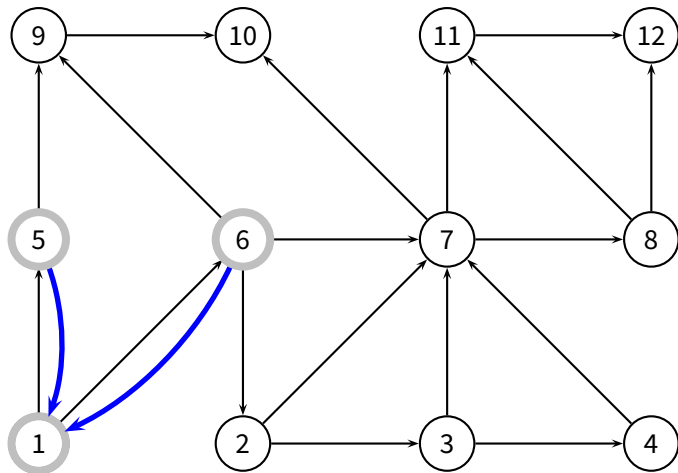


Example

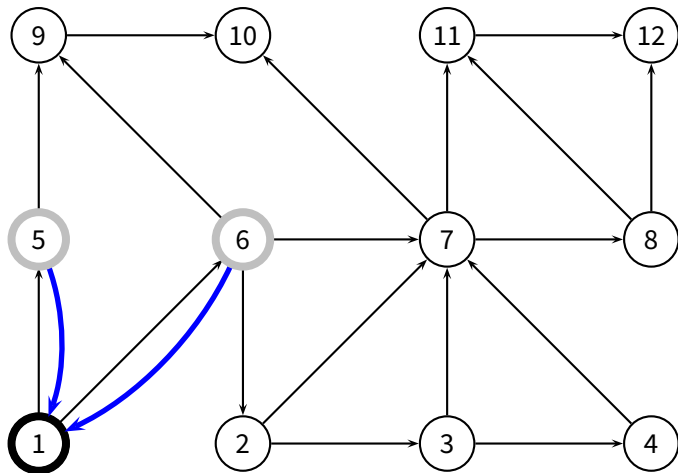


Example

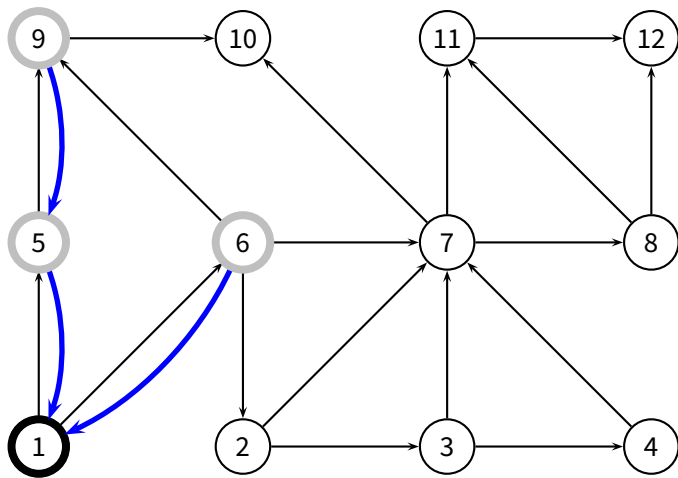




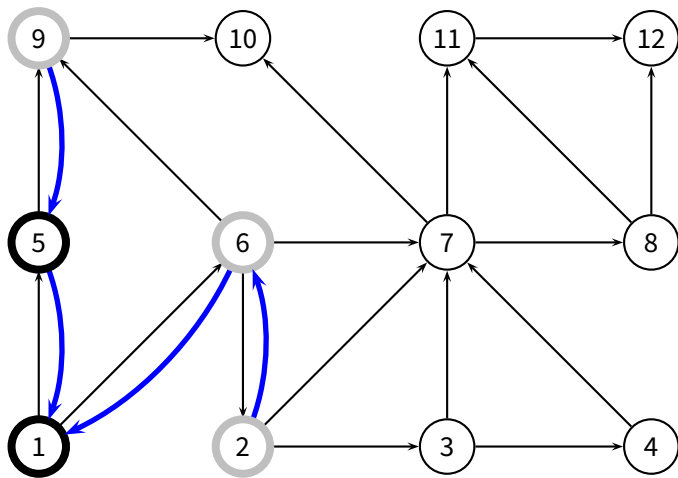
Example



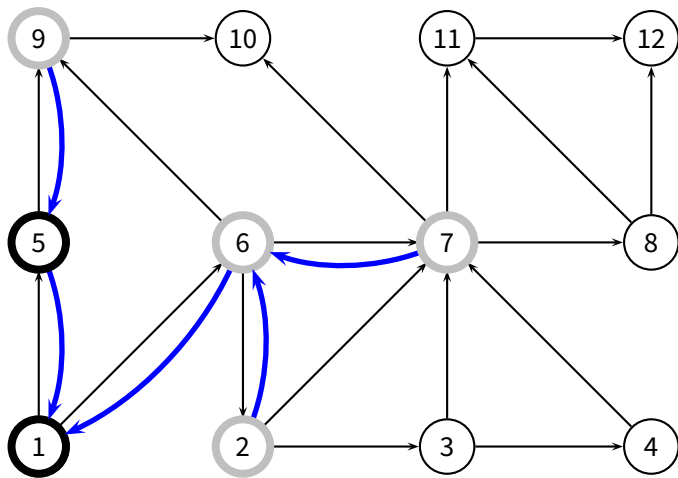
Example



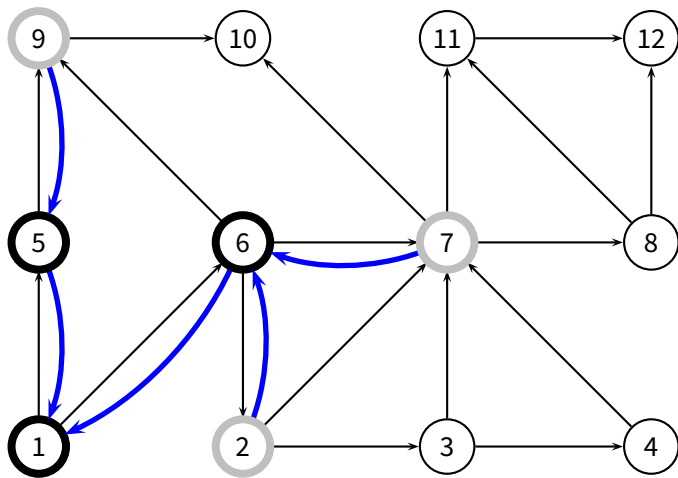
Example



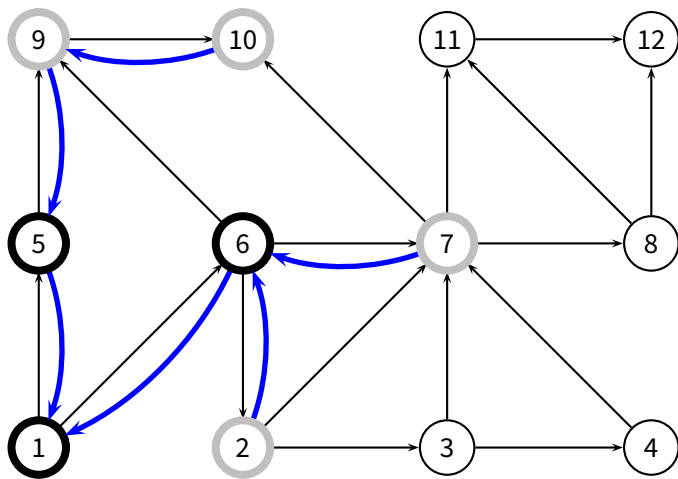
Example



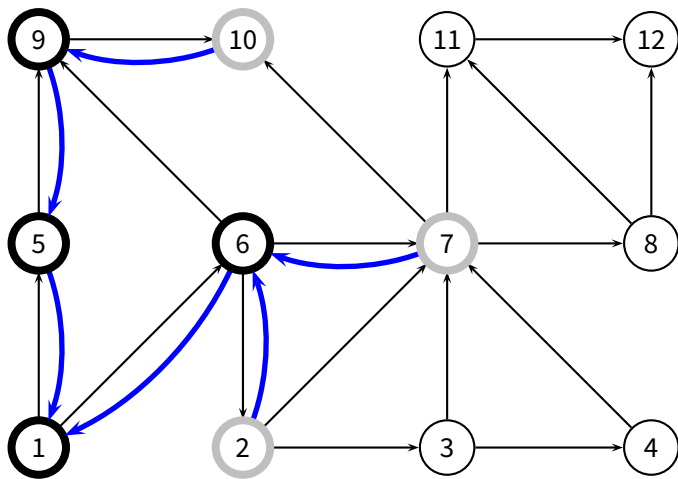
Example



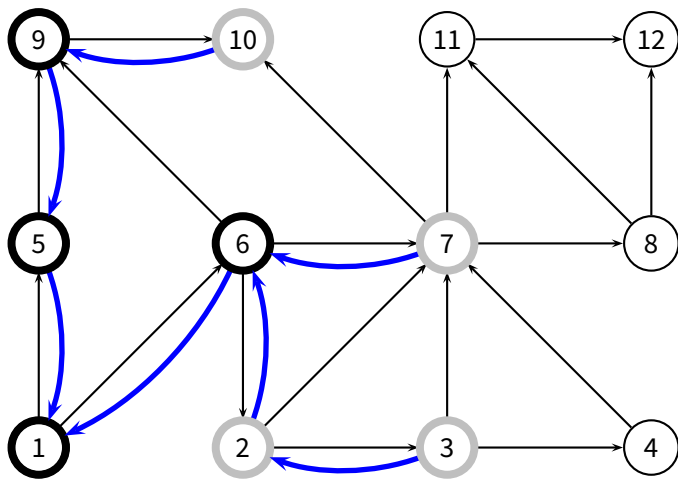
Example



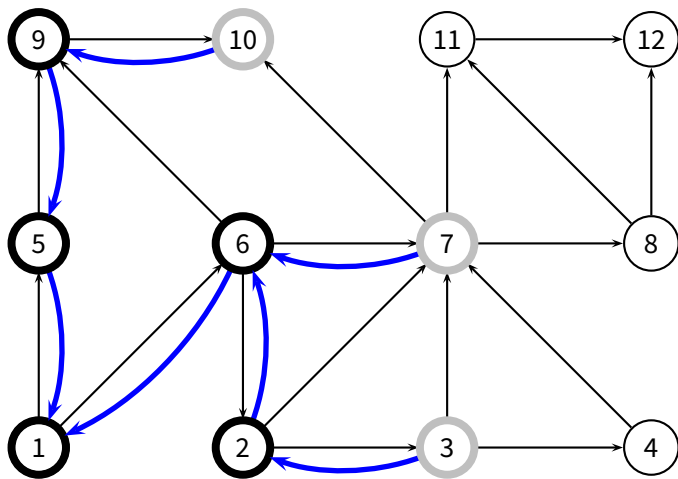
Example



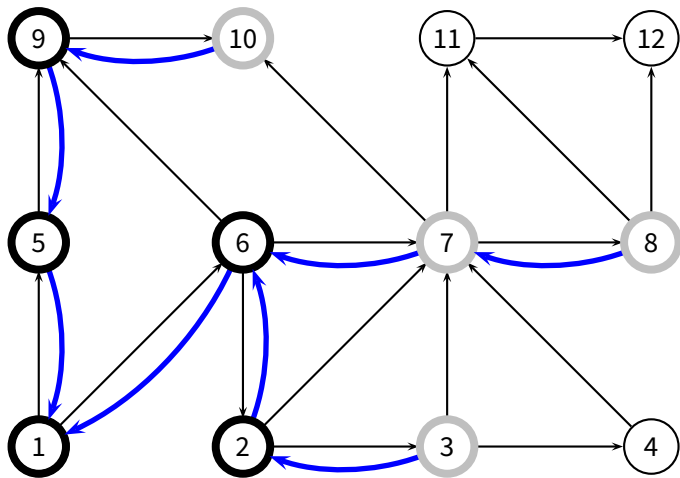
Example



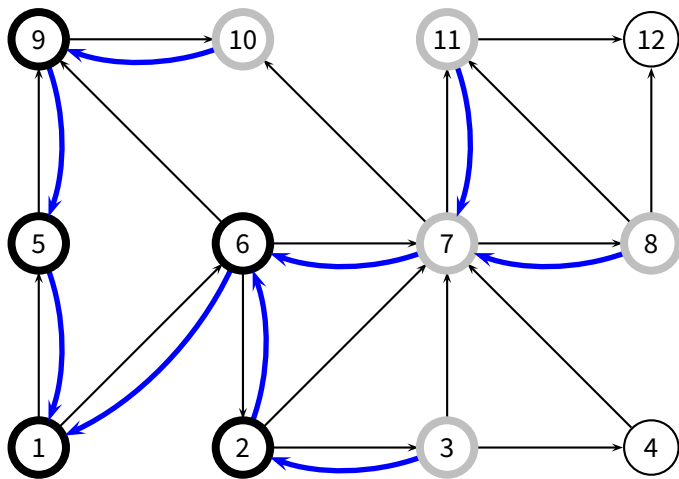
Example



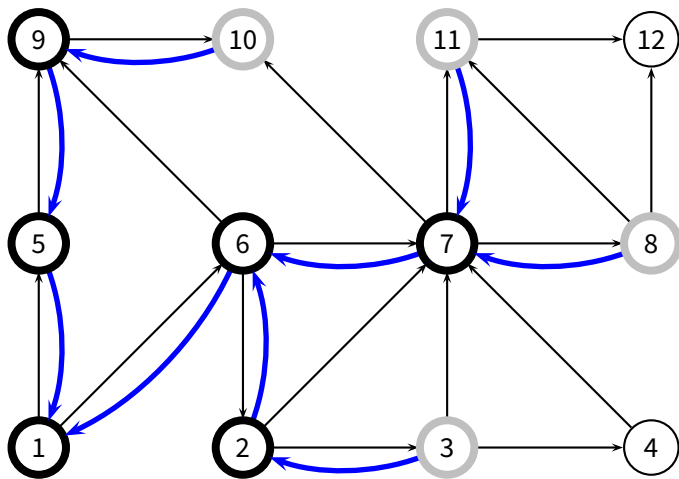
Example



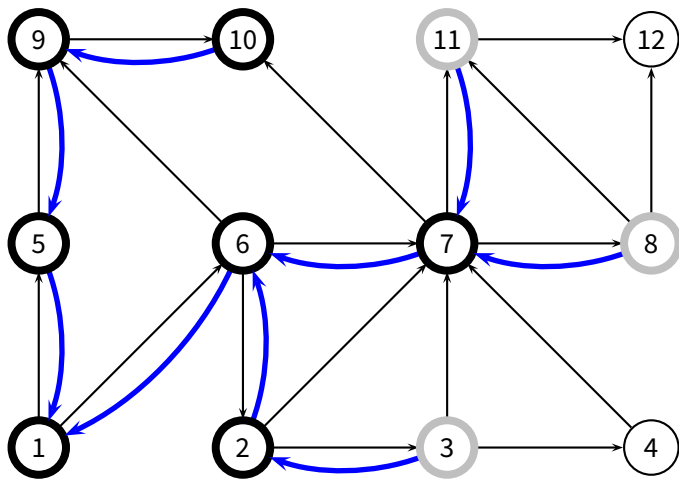
Example



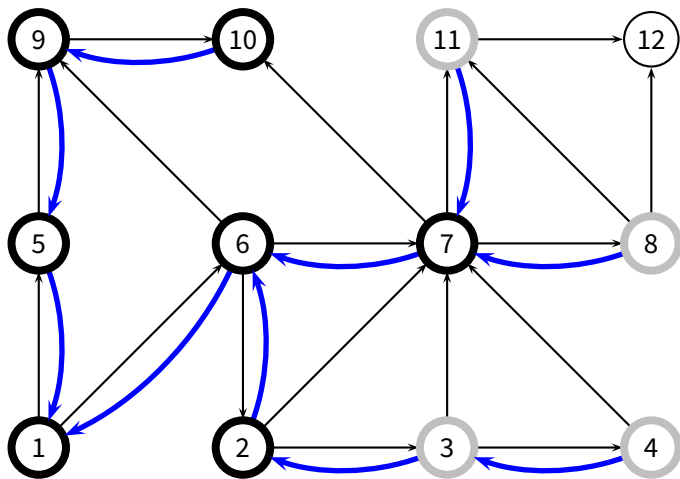
Example



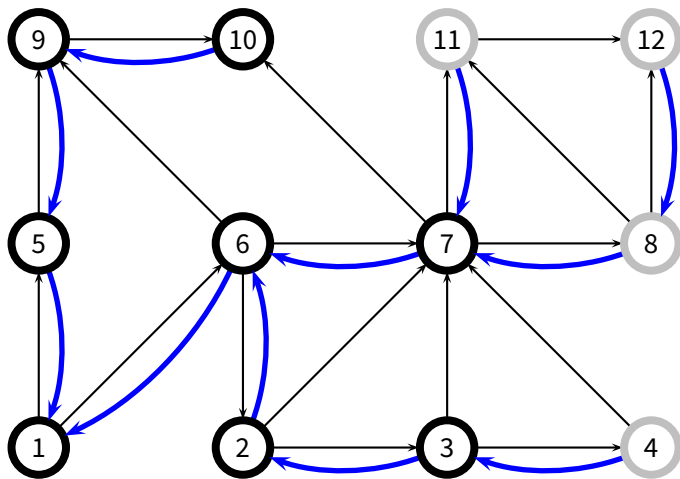
Example



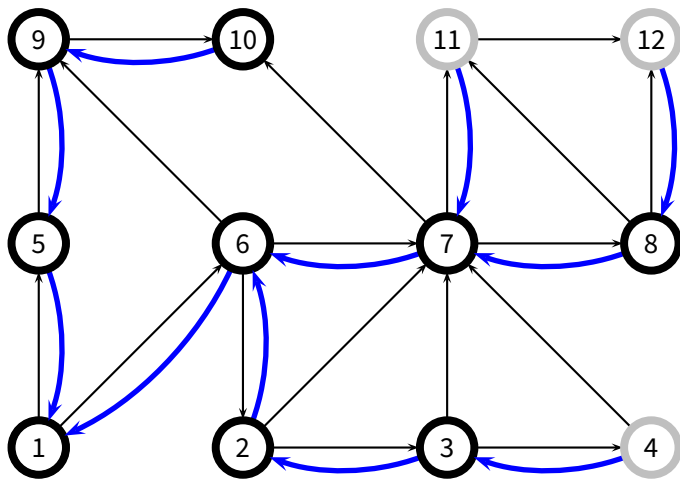
Example



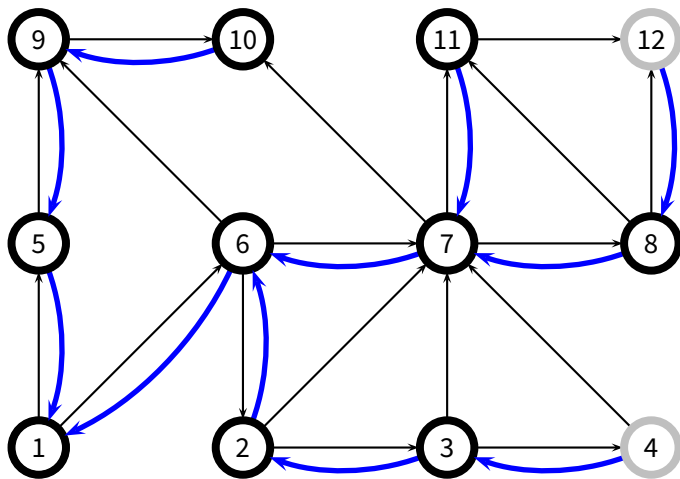
Example



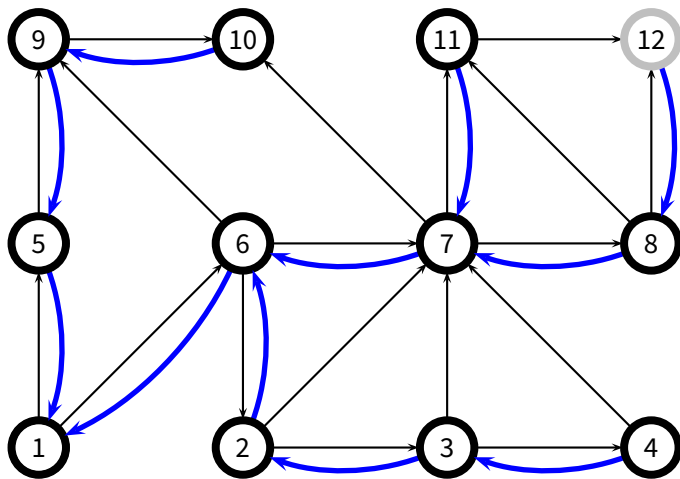
Example



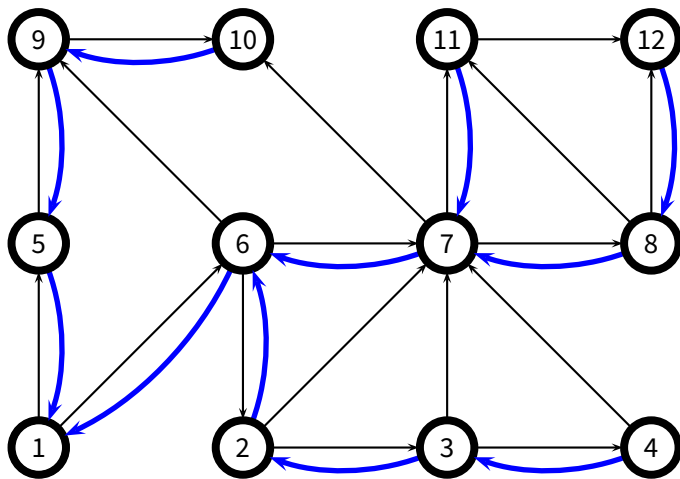
Example



Example



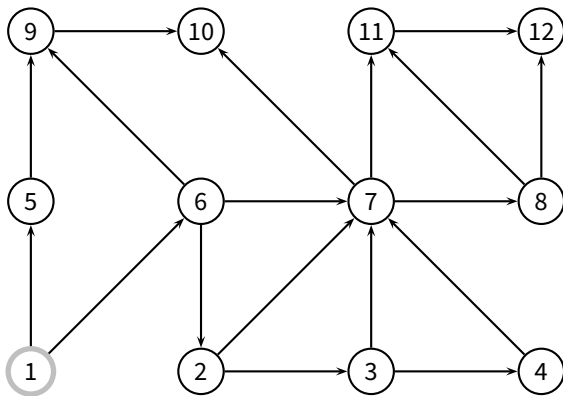
Example



BFS Algorithm

BFS(G, s)

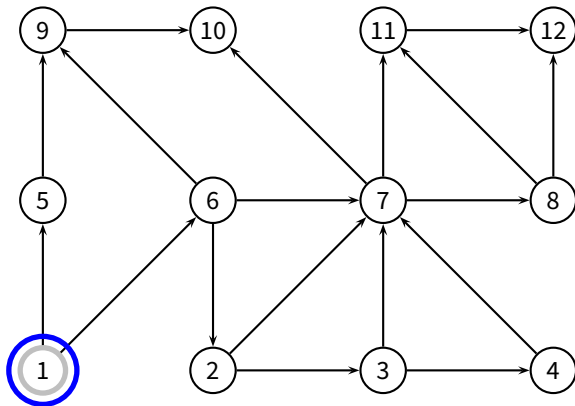
```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



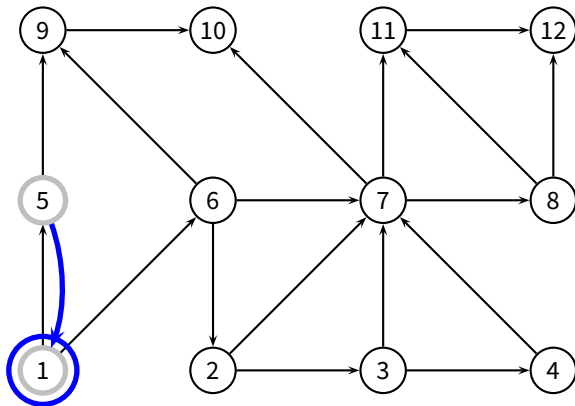
$u = 1$

$Q = \emptyset$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



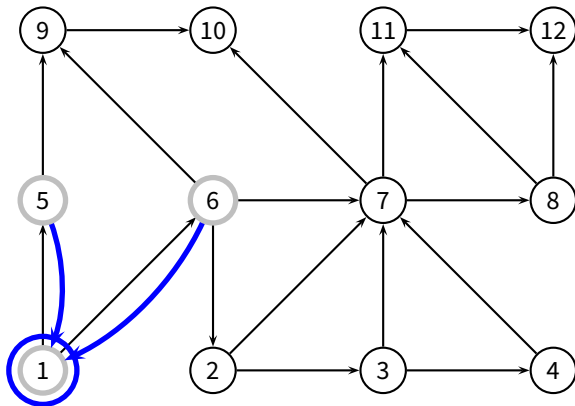
$u = 1$

$Q = \{5\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



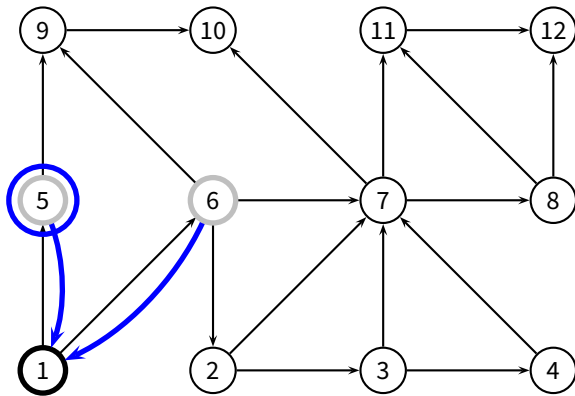
$u = 1$

$Q = \{5, 6\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



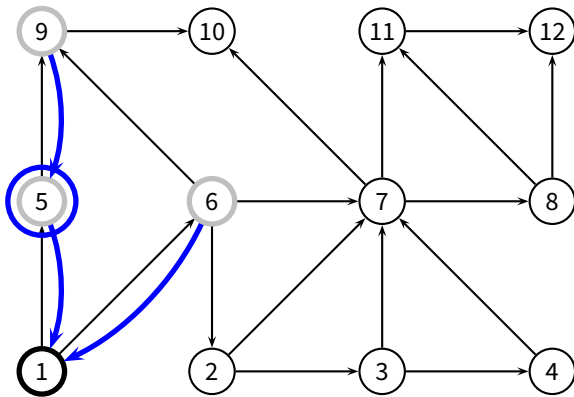
$u = 5$

$Q = \{6\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



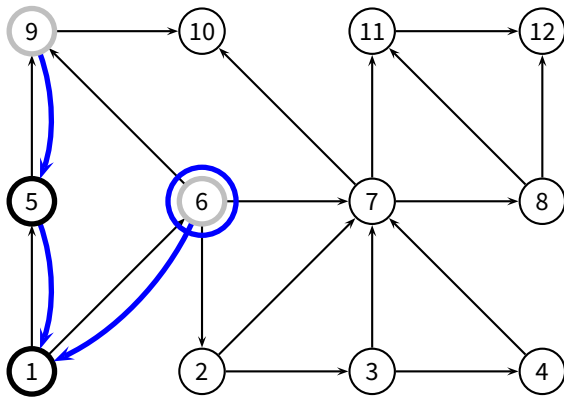
$u = 5$

$Q = \{6, 9\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



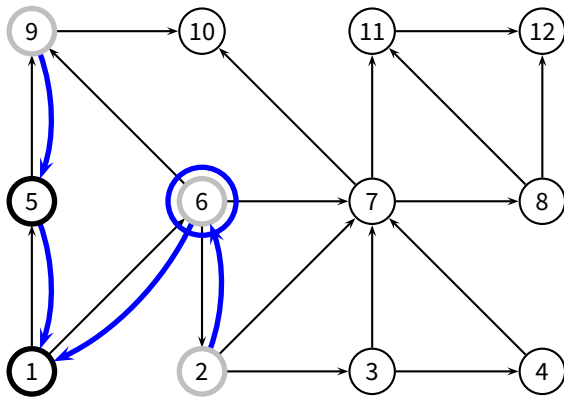
$u = 6$

$Q = \{9\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



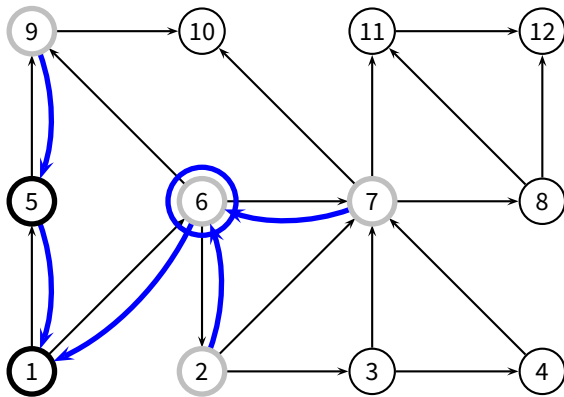
$u = 6$

$Q = \{9, 2, 7\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



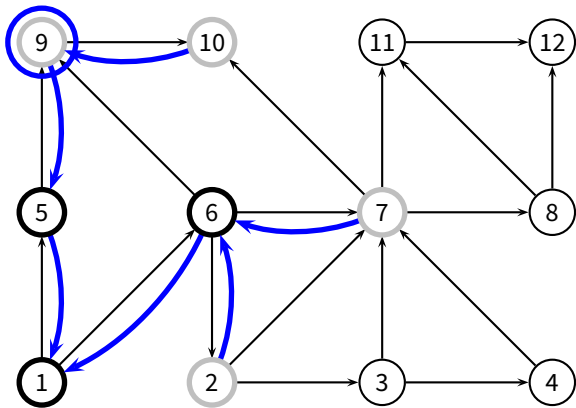
$u = 6$

$Q = \{9, 2, 7\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



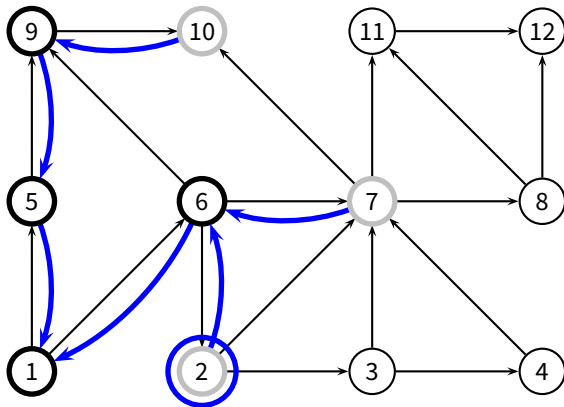
$u = 9$

$Q = \{2, 7, 10\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



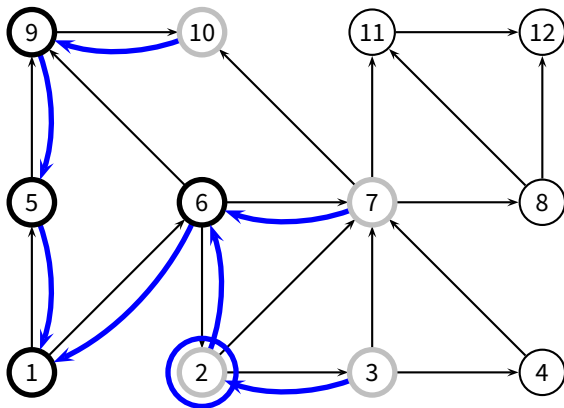
$u = 2$

$Q = \{7, 10\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



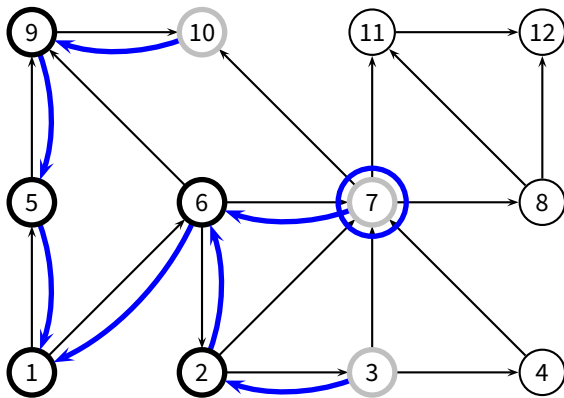
$u = 2$

$Q = \{7, 10, 3\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



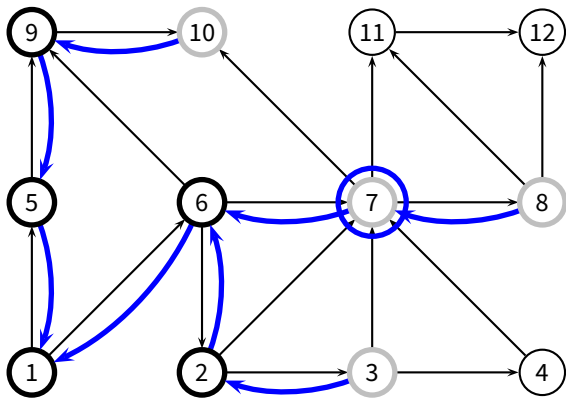
$u = 7$

$Q = \{10, 3\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



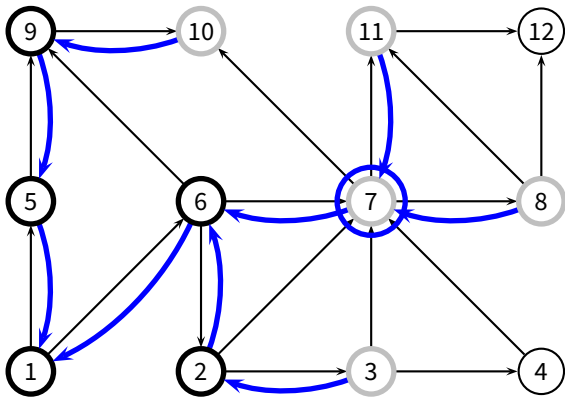
$u = 7$

$Q = \{10, 3, 8\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



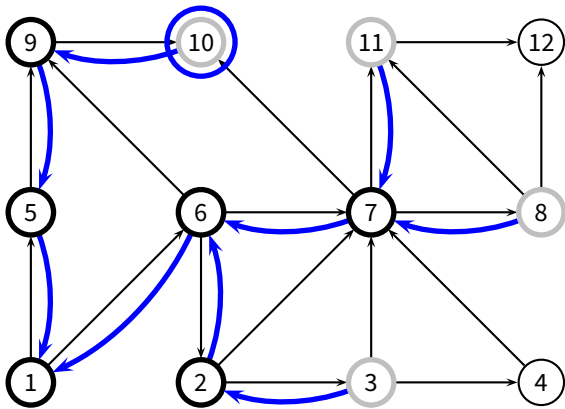
$u = 7$

$Q = \{10, 3, 8, 11\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



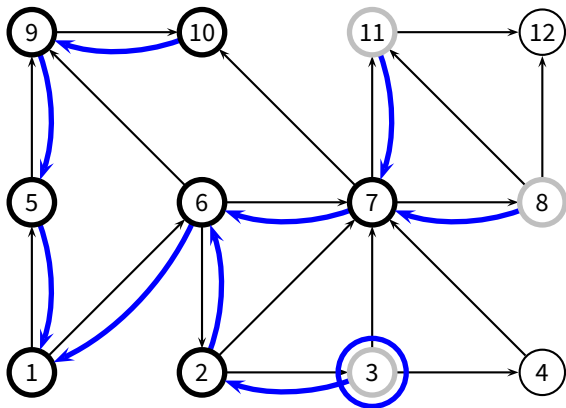
$u = 10$

$Q = \{3, 8, 11\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



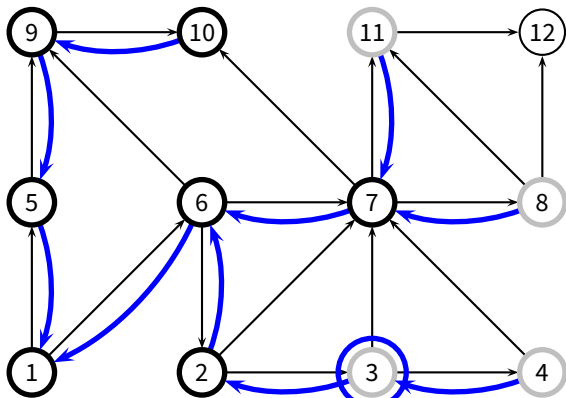
$u = 3$

$Q = \{8, 11\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



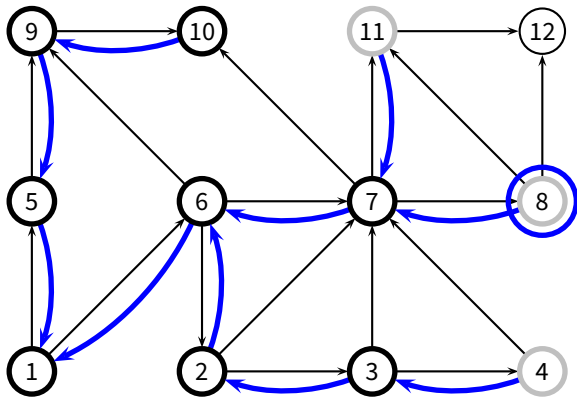
$u = 3$

$Q = \{8, 11, 4\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



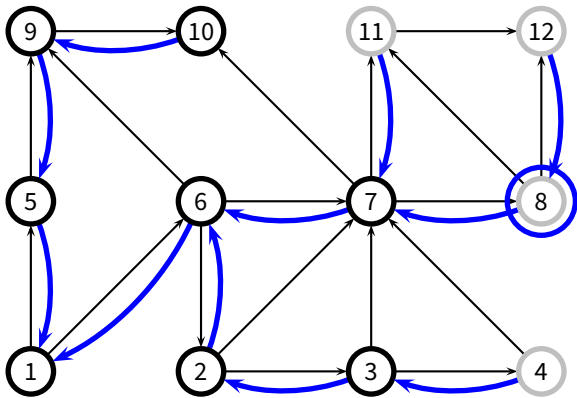
$u = 8$

$Q = \{11, 4\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



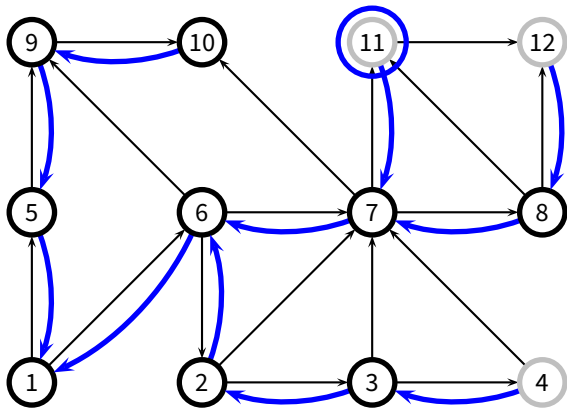
$u = 8$

$Q = \{11, 4, 12\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



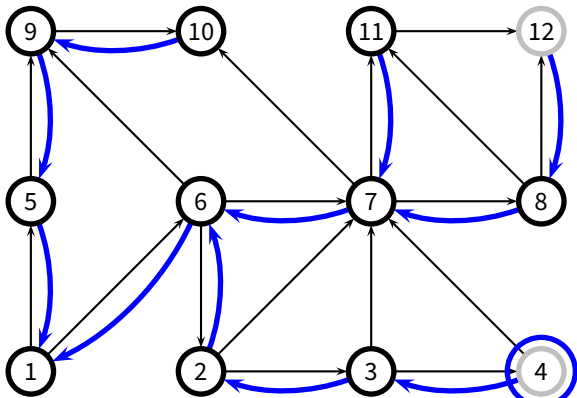
$u = 11$

$Q = \{4, 12\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



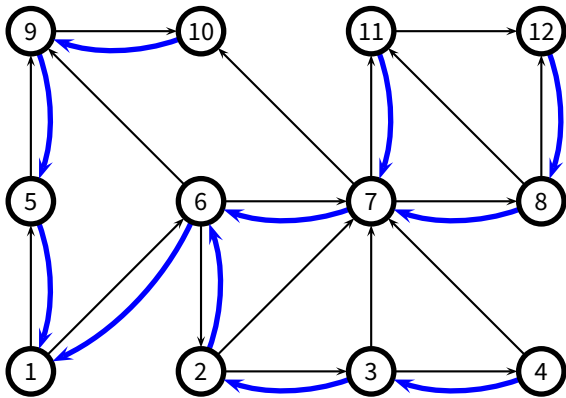
$u = 4$

$Q = \{12\}$

BFS Algorithm

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```



BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in Adj[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```

- We enqueue a vertex only if it is white, and we immediately color it gray; thus, we enqueue every vertex *at most once*

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```

- We enqueue a vertex only if it is white, and we immediately color it gray; thus, we enqueue every vertex *at most once*
- So, the (dequeue) while loop executes $O(|V|)$ times

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in \text{Adj}[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```

- We enqueue a vertex only if it is white, and we immediately color it gray; thus, we enqueue every vertex *at most once*
- So, the (dequeue) while loop executes $O(|V|)$ times
- For each vertex u , the inner loop executes $\Theta(|E_u|)$, for a total of $O(|E|)$ steps

BFS(G, s)

```
1  for each vertex  $u \in V(G) \setminus \{s\}$ 
2       $color[u] = \text{WHITE}$ 
3       $d[u] = \infty$ 
4       $\pi[u] = \text{NIL}$ 
5   $color[s] = \text{GRAY}$ 
6   $d[s] = 0$ 
7   $\pi[s] = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in Adj[u]$ 
13         if  $color[v] == \text{WHITE}$ 
14              $color[v] = \text{GRAY}$ 
15              $d[v] = d[u] + 1$ 
16              $\pi[v] = u$ 
17             ENQUEUE( $Q, v$ )
18      $color[u] = \text{BLACK}$ 
```

- We enqueue a vertex only if it is white, and we immediately color it gray; thus, we enqueue every vertex *at most once*
- So, the (dequeue) while loop executes $O(|V|)$ times
- For each vertex u , the inner loop executes $\Theta(|E_u|)$, for a total of $O(|E|)$ steps
- So, $O(|V| + |E|)$

- Immediately follow the links of the most recently-visited vertex, then backtrack when you reach a dead-end
 - ▶ i.e., backtrack when the current vertex has no more adjacent vertices that have not yet been visited

- Immediately follow the links of the most recently-visited vertex, then backtrack when you reach a dead-end
 - ▶ i.e., backtrack when the current vertex has no more adjacent vertices that have not yet been visited
- *Input: $G = (V, E)$*
 - ▶ explores the graph, touching *all vertices*

- Immediately follow the links of the most recently-visited vertex, then backtrack when you reach a dead-end
 - ▶ i.e., backtrack when the current vertex has no more adjacent vertices that have not yet been visited
- *Input: $G = (V, E)$*
 - ▶ explores the graph, touching *all vertices*
 - ▶ produces a ***depth-first forest***, consisting of all the ***depth-first trees*** defined by the DFS exploration

- Immediately follow the links of the most recently-visited vertex, then backtrack when you reach a dead-end
 - ▶ i.e., backtrack when the current vertex has no more adjacent vertices that have not yet been visited
- *Input: $G = (V, E)$*
 - ▶ explores the graph, touching *all vertices*
 - ▶ produces a **depth-first forest**, consisting of all the **depth-first trees** defined by the DFS exploration
 - ▶ associates **two time-stamps** to each vertex
 - ▶ $d[u]$ records when u is first discovered
 - ▶ $f[u]$ records when DFS finishes examining u 's edges, and therefore backtracks from u

DFS(G)

```
1 for each vertex  $u \in V(G)$ 
2    $color[u] = WHITE$ 
3    $\pi[u] = NIL$ 
4  $time = 0$  // “global” variable
5 for each vertex  $u \in V(G)$ 
6   if  $color[u] == WHITE$ 
7     DFS-VISIT( $u$ )
```

DFS-VISIT(u)

```
1  $color[u] = GREY$ 
2  $time = time + 1$ 
3  $d[u] = time$ 
4 for each  $v \in Adj[u]$ 
5   if  $color[v] == WHITE$ 
6      $\pi[v] = u$ 
7     DFS-VISIT( $v$ )
8  $color[u] = BLACK$ 
9  $time = time + 1$ 
10  $f[u] = time$ 
```


- The loop in **DFS-VISIT**(u) (lines 4–7) accounts for $\Theta(|E_u|)$

- The loop in **DFS-VISIT**(u) (lines 4–7) accounts for $\Theta(|E_u|)$
- We call **DFS-VISIT**(u) *once* for each vertex u
 - ▶ either in **DFS**, or recursively in **DFS-VISIT**
 - ▶ because we call it only if $color[u] = \text{WHITE}$, but then we immediately set $color[u] = \text{GREY}$

- The loop in **DFS-VISIT**(u) (lines 4–7) accounts for $\Theta(|E_u|)$
- We call **DFS-VISIT**(u) *once* for each vertex u
 - ▶ either in **DFS**, or recursively in **DFS-VISIT**
 - ▶ because we call it only if $color[u] = \text{WHITE}$, but then we immediately set $color[u] = \text{GREY}$
- So, the overall complexity is $\Theta(|V| + |E|)$

Applications of DFS: Topological Sort

Applications of DFS: Topological Sort

- **Problem:** (topological sort)

Given a *directed acyclic graph* (DAG)

- ▶ find an ordering of vertices such that you only end up with *forward links*

Applications of DFS: Topological Sort

■ **Problem:** (topological sort)

Given a *directed acyclic graph* (DAG)

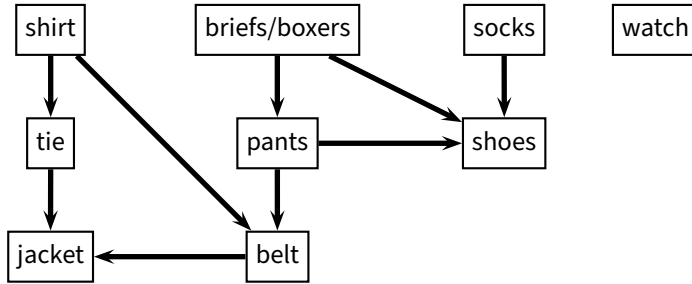
- ▶ find an ordering of vertices such that you only end up with *forward links*

■ **Example:** dependencies in software packages

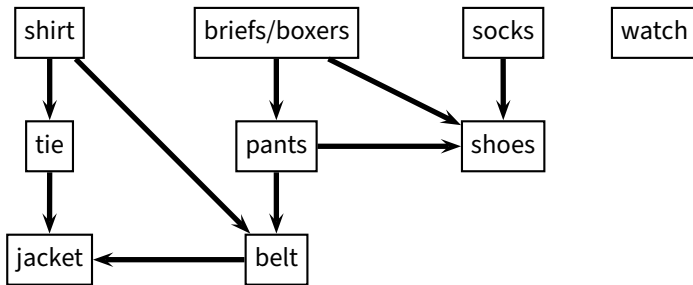
- ▶ find an installation order for a set of software packages
- ▶ such that every package is installed only after all the packages it depends on

Topological Sort Algorithm

Topological Sort Algorithm



Topological Sort Algorithm



TOPOLOGICAL-SORT (G)

- 1 **DFS**(G)
- 2 output V sorted in reverse order of $f[\cdot]$